A Context-Driven Approach for Guiding Agile Adoption

Ayed, Hajer; Vanderose, Benoît; Habra, Naji

Published in:
ICSEA 2015, The Tenth International Conference on Software Engineering Advances

Publication date:
2015

Document Version
Peer reviewed version

ICSEA 2015

The Tenth International Conference on Software Engineering Advances

ISBN: 978-1-61208-438-1

November 15 - 20, 2015

Barcelona, Spain

ICSEA 2015 Editors

Roy Oberhauser, Aalen University, Germany

Luigi Lavazza, Università dell'Insubria - Varese, Italy

Herwig Mannaert, University of Antwerp, Belgium

Stephen Clyde, Utah State University, USA

ICSEA 2015

Forward

The Tenth International Conference on Software Engineering Advances (ICSEA 2015), held on November 15 - 20, 2015 in Barcelona, Spain, continued a series of events covering a broad spectrum of software-related topics.

The conference covered fundamentals on designing, implementing, testing, validating and maintaining various kinds of software. The tracks treated the topics from theory to practice, in terms of methodologies, design, implementation, testing, use cases, tools, and lessons learnt. The conference topics covered classical and advanced methodologies, open source, agile software, as well as software deployment and software economics and education.

The conference had the following tracks:

- Advances in fundamentals for software development
- Advanced mechanisms for software development
- Advanced design tools for developing software
- Software engineering for service computing (SOA and Cloud)
- Advanced facilities for accessing software
- Software performance
- Software security, privacy, safeness
- Advances in software testing
- Specialized software advanced applications
- Web Accessibility
- Open source software
- Agile and Lean approaches in software engineering
- Software deployment and maintenance
- Software engineering techniques, metrics, and formalisms
- Software economics, adoption, and education
- Business technology
- Improving productivity in research on software engineering

Similar to the previous edition, this event continued to be very competitive in its selection process and very well perceived by the international software engineering community. As such, it is attracting excellent contributions and active participation from all over the world. We were very pleased to receive a large amount of top quality contributions.

We take here the opportunity to warmly thank all the members of the ICSEA 2015 technical program committee as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and efforts to contribute to the ICSEA 2015. We truly believe that thanks to all these efforts, the final conference program consists of top quality contributions.

This event could also not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the ICSEA 2015 organizing committee for their help in handling the logistics and for their work that is making this professional meeting a success.

We hope the ICSEA 2015 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in software engineering research. We also hope Barcelona provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city.

ICSEA 2015 Advisory Chairs

Herwig Mannaert, University of Antwerp, Belgium
Mira Kajko-Mattsson, Stockholm University & Royal Institute of Technology, Sweden
Luigi Lavazza, Università dell'Insubria - Varese, Italy
Roy Oberhauser, Aalen University, Germany
Elena Troubitsyna, Åbo Akademi University, Finland
Davide Tosi, Università dell'Insubria - Como, Italy
Luis Fernandez-Sanz, Universidad de Alcala, Spain
Michael Gebhart, iterate GmbH, Germany

ICSEA 2015 Research Institute Liaison Chairs

Oleksandr Panchenko, Hasso Plattner Institute for Software Systems Engineering - Potsdam, Germany
Teemu Kanstrén, VTT Technical Research Centre of Finland - Oulu, Finland
Osamu Takaki, Gunma University, Japan
Georg Buchgeher, Software Competence Center Hagenberg GmbH, Austria

ICSEA 2015 Industry/Research Chairs

Herman Hartmann, University of Groningen, The Netherlands
Hongyu Pei Breivold, ABB Corporate Research, Sweden

ICSEA 2015 Special Area Chairs

Formal Methods
Paul J. Gibson, Telecom & Management SudParis, France

Testing and Validation
Florian Barth, University of Mannheim, Germany

Web Accessibility
Adriana Martin, National University of Austral Patagonia (UNPA), Argentina

Software engineering for service computing
Muthu Ramachandran, Leeds Beckett University, UK

ICSEA 2015 Publicity Chairs

Sébastien Salva, University of Auvergne, Clermont-Ferrand, France

ICSEA 2015

Committee

ICSEA Advisory Chairs

Herwig Mannaert, University of Antwerp, Belgium
Mira Kajko-Mattsson, Stockholm University & Royal Institute of Technology, Sweden
Luigi Lavazza, Università dell'Insubria - Varese, Italy
Roy Oberhauser, Aalen University, Germany
Elena Troubitsyna, Åbo Akademi University, Finland
Davide Tosi, Università dell'Insubria - Como, Italy
Luis Fernandez-Sanz, Universidad de Alcala, Spain
Michael Gebhart, iteratec GmbH, Germany

ICSEA 2015 Research Institute Liaison Chairs

Oleksandr Panchenko, Hasso Plattner Institute for Software Systems Engineering - Potsdam, Germany
Teemu Kanstrén, VTT Technical Research Centre of Finland - Oulu, Finland
Osamu Takaki, Gunma University, Japan
Georg Buchgeher, Software Competence Center Hagenberg GmbH, Austria

ICSEA 2015 Industry/Research Chairs

Herman Hartmann, University of Groningen, The Netherlands
Hongyu Pei Breivold, ABB Corporate Research, Sweden

ICSEA 2015 Special Area Chairs

Formal Methods
Paul J. Gibson, Telecom & Management SudParis, France

Testing and Validation
Florian Barth, University of Mannheim, Germany

Web Accessibility
Adriana Martin, National University of Austral Patagonia (UNPA), Argentina

Software engineering for service computing
Muthu Ramachandran, Leeds Beckett University, UK

ICSEA 2015 Publicity Chairs

Sébastien Salva, University of Auvergne, Clermont-Ferrand, France

ICSEA 2015 Technical Program Committee

Shahliza Abd Halim, Universiti of Technologi Malaysia (UTM) - Skudai, Malaysia
Mohammad Abdallah, Al-Zaytoonah University of Jordan, Jordan
Adla Abdelkader, University of Oran, Algeria
Muhammad Ovais Ahmad, University of Oulu, Finland
Moataz A. Ahmed, King Fahd University of Petroleum & Minerals – Dhahran, Saudi Arabia
Syed Nadeem Ahsan, TU-Graz, Austria
Mehmet Aksit, University of Twente, Netherlands
Ahmed Al-Moayed, Hochschule Furtwangen University, Germany
Azadeh Alebrahim, University of Duisburg-Essen, Germany
Mamdouh Alenezi, Prince Sultan University - Riyadh, Saudi Arabia
Basem Y. Alkazemi, Umm Al-Qura University, Saudi Arabia
Mohammad Alshayeb, King Fahd University of Petroleum and Minerals, Saudi Arabia
Zakarya A. Alzamil, King Saud University, Saudi Arabia
Vincenzo Ambriola, Università di Pisa, Italy
Jose Andre Dorigan, State University of Maringa, Brazil
Buzzi Andreas, Credit Suisse AG – Zürich, Switzerland
Andreas S. Andreou, Cyprus University of Technology - Limassol, Cyprus
Maria Anjum, Durham University, UK
Paulo Asterio de Castro Guerra, Tapijara Programação de Sistemas Ltda. - Lambari, Brazil
Colin Atkinson, University of Mannheim, Germany
Marco Autili, University of L'Aquila, Italy
Robert Azarbod, Oracle Corporation, USA
Thomas Baar, Hochschule für Technik und Wirtschaft (HTW) Berlin, Germany
Gilbert Babin, HEC Montréal, Canada
Muneera Bano, International Islamic University - Islamabad, Pakistan
Fernando Sérgio Barbosa, Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco, Portugal
Jorge Barreiros, CITI/UNL: Center of Informatics and Information Technology - UNL || ISEC/IPC: ISEC - Polytechnic Institute of Coimbra, Portugal
Florian Barth, University of Mannheim, Germany
Gabriele Bavota, University of Salerno, Italy
Noureddine Belkhatir, University of Grenoble, France
Simona Bernardi, Centro Universitario de la Defensa / Academia General Militar - Zaragoza, Spain
Jorge Bernardino, Polytechnic Institute of Coimbra - ISEC-CISUC, Portugal
Ateet Bhalla, Independent Consultant, India
Celestina Bianco, Systelab Technologies - Barcelona, Spain
Christian Bird, University of California, USA
Kenneth Boness, Reading University, UK
Mina Boström Nakicenovic, Sungard Front Arena, Stockholm, Sweden
M. Boukala-Ioualalen, University of Science and Technology Houari Boumediene, Algeria
Fernando Brito e Abreu, Instituto Universitário de Lisboa (ISCTE-IUL), Portugal
Hongyu Pei Breivold, ABB Corporate Research, Sweden
Manfred Broy, Technische Universität München, Germany
Georg Buchgeher, Software Competence Center Hagenberg GmbH, Austria
Thomas Buchmann, Universität Bayreuth, Germany
Lucas Bueno Ruas de Oliveira, University of São Paulo (ICMC/USP), Brazil

Claus Pahl, Dublin City University, Ireland
Marcos Palacios, University of Oviedo, Spain
Fabio Palomba, University of Salerno, Italy
Päivi Parviainen, VTT, Software Technologies Center, Finland
Aljosa Pasic, ATOS Research, Spain
Fabrizio Pastore, University of Milano - Bicocca, Italy
Asier Perallos, University of Deusto, Spain
Óscar Pereira, University of Aveiro, Portugal
Beatriz Pérez Valle, University of La Rioja, Spain
David Pheanis, Arizona State University, USA
Pasqualina Potena, University of Alcalá, Spain
Christian Prehofer, Kompetenzfeldleiter Adaptive Kommunikationssysteme / Fraunhofer-Einrichtung für Systeme der Kommunikationstechnik ESK – München, Germnay
Abdallah Qusef, University of Salerno, Italy
Salman Rafiq, Fraunhofer Institute for Embedded Systems and Communication Technologies, Germany
Claudia Raibulet, Università degli Studi di Milano-Bicocca, Italy
Muthu Ramachandran, Leeds Beckett University, UK
Amar Ramdane-Cherif, University of Versailles, France
Raman Ramsin, Sharif University of Technology, Iran
Gianna Reggio, DIBRIS - Università di Genova, Italy
Zhilei Ren, Dalian University of Technology, China
Hassan Reza, University of North Dakota - School of Aerospace, USA
Samir Ribic, University of Sarajevo, Bosnia and Herzegovina
Elvinia Riccobene, University of Milan, Italy
Daniel Riesco, National University of San Luis, Argentina
Michele Risi, University of Salerno, Italy
Gabriela Robiolo, Universidad Austral, Argentina
Oliveto Rocco, University of Molise, Italy
Rodrigo G. C. Rocha, Federal Rural University of Pernambuco, Brazil
Daniel Rodríguez, University of Alcalá, Madrid, Spain
María Luisa Rodríguez Almendros, Universidad de Granada, Spain
Siegfried Rouvrais, Institut Mines Telecom Bretagne, France
Suman Roychoudhury, Tata Consultancy Services, India
Mercedes Ruiz Carreira, Universidad de Cádiz, Spain
Alessandra Russo, Imperial College London, UK
Mehrdad Saadatmand, Mälardalen University / Alten AB, Sweden
Krzysztof Sacha, Warsaw University of Technology, Poland
Francesca Saglietti, University of Erlangen-Nuremberg, Germany
Sébastien Salva, LIMOS-CNRS / Auvergne University / IUT d'Aubière, France
Maria-Isabel Sanchez-Segura, Carlos III University of Madrid, Spain
Luca Santillo, Agile Metrics, Italy
Gaetana Sapienza, ABB Corporate Research, Sweden
Federica Sarro, University College London, UK
Patrizia Scandurra, University of Bergamo - Dalmine, Italy
Giuseppe Scanniello, Università degli Studi della Basilicata - Potenza, Italy
Christelle Scharff, Pace University, USA
Klaus Schmid, University of Hildesheim, Germany
Felix Schwägerl, University of Bayreuth, Germany

Bran Selic, Malina Software, Canada
Fereidoon Shams, Shahid Beheshti University, Iran
Fernando Selleri Silva, Mato Grosso State University (UNEMAT), Brazil
István Siket, University of Szeged, Hungary
Abu Bakar Md Sultan, Universiti Putra Malaysia, Malaysia
Sidra Sultana, National University of Sciences and Technology, Pakistan
Lijian Sun, Chinese Academy of Surveying & Mapping, China
Mahbubur R. Syed, Minnesota State University – Mankato, USA
Davide Taibi, Free University of Bozen, Italy
Osamu Takaki, Gunma University, Japan
Giordano Tamburrelli, Università della Svizzera Italiana (USI), Swizterland
Wasif Tanveer, University of Engineering and Technology - Lahore, Pakistan
Nebojša Taušan, University of Oulu, Finland
Pierre Tiako, Langston University, USA
Maarit Tihinen, VTT Technical Research Centre of Finland - Oulu, Finland
Massimo Tivoli, University of L'Aquila, Italy
Maria Tortorella, University of Sannnio - Benevento Italy
Davide Tosi, Università degli studi dell'Insubria - Varese, Italy
Peter Trapp, Ingolstadt, Germany
Elena Troubitsyna, Åbo Akademi University, Finland
Mariusz Trzaska, Polish-Japanese Academy of Information Technology, Poland
George A. Tsihrintzis, University of Piraeus, Greece
Masateru Tsunoda, Kinki University, Japan
Henry Tufo, University of Colorado at Boulder, USA
Javier Tuya, Universidad de Oviedo - Gijón, Spain
Andreas Ulrich, Siemens AG, Germany
Christelle Urtado, LGI2P / Ecole des Mines d'Alès - Nîmes, France
Dieter Van Nuffel, University of Antwerp, Belgium
Timo Vepsäläinen, Tampere University of Technology, Finland
Laszlo Vidacs, Hungarian Academy of Sciences, Hungary
Tanja Vos, Universidad Politécnica de Valencia, Spain
Stefan Wagner, University of Stuttgart, Germany
Hironori Washizaki, Waseda University, Japan
Stefan Wendler, Ilmenau University of Technology, Germany
Agnes Werner-Stark, University of Pannonia, Hungary
Norman Wilde, University of West Florida, USA
Andreas Winter, Carl von Ossietzky University, Germany
Victor Winter, University of Nebraska-Omaha, USA
Martin Wojtczyk, Technische Universität München, Germany & Cubotix, USA
Haibo Yu, Shanghai Jiao Tong University, China
Elisa Yumi Nakagawa, University of São Paulo (USP), Brazil
Saad Zafar, Riphah International University - Islamabad, Pakistan
Amir Zeid, American University of Kuwait, Kuwait
Michal Zemlicka, University of Finance and Administration, Czech Republic
Gefei Zhang, Celonis GmbH, Germany
Qiang Zhu, The University of Michigan - Dearborn, USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Patterns for Specifying Bidirectional Transformations in UML-RSDS

K. Lano,
S. Yassipour-Tehrani
Dept. of Informatics
King's College London
London, UK
Email: kevin.lano@kcl.ac.uk,
s.yassipour-tehrani@kcl.ac.uk

S. Kolahdouz-Rahimi
Dept of Software Engineering
University of Isfahan
Isfahan, Iran
Email: sh.rahimi@eng.ui.ac.ir

***Abstract*—In this paper, we identify model transformation specification and design patterns, which support the property of transformation bidirectionality: the ability of a single specification to be applied either as a source-to-target transformation or as a target-to-source transformation. In contrast to previous work on bidirectional transformations (bx), we identify the important role of transformation *invariants* in the derivation of reverse transformations, and show how patterns and invariants can be used to give a practical means of defining bx in the UML-RSDS transformation language.**

Keywords — Bidirectional transformations; transformation design patterns; UML-RSDS

I. INTRODUCTION

Bidirectional transformations (bx) are considered important in a number of transformation scenarios:

- Maintaining consistency between two models which may both change, for example, if a UML class diagram and corresponding synthesised Java code both need to be maintained consistently with each other, in order to implement *round-trip engineering* for model-driven development.
- Where a mapping between two languages may need to be operated in either direction for different purposes, for example, to represent behavioural models as either Petri Nets or as state machines [12].
- Where inter-conversion between two different representations is needed, such as two alternative formats of electronic health record [3].

Design patterns have become an important tool in software engineering, providing a catalogue of 'best practice' solutions to design problems in software [7]. Patterns for model transformations have also been identified [14], but patterns specifically for bx have not been defined.

In this paper, we show how bx patterns can be used to obtain a practical approach for bx using the UML-RSDS language [11].

Section II defines the concept of a bx. Section V describes related work. Section III describes UML-RSDS and transformation specification in UML-RSDS. Section IV gives a catalogue of bx patterns for UML-RSDS, with examples. Section VI gives a conclusion.

II. CRITERIA FOR BIDIRECTIONALITY

Bidirectional transformations are characterised by a binary relation $R : SL \leftrightarrow TL$ between a source language (metamodel) SL and a target language TL. $R(m,n)$ holds for a pair of models m of SL and n of TL when the models consist of data which corresponds under R. It should be possible to automatically derive from the definition of R both forward and reverse transformations

$$R^{\rightarrow} : SL \times TL \rightarrow TL \qquad R^{\leftarrow} : SL \times TL \rightarrow SL$$

which aim to establish R between their first (respectively second) and their result target (respectively source) models, given both existing source and target models.

Stevens [16] has identified two key conditions which bidirectional model transformations should satisfy:

1) Correctness: the forward and reverse transformations derived from a relation R do establish R: $R(m, R^{\rightarrow}(m,n))$ and $R(R^{\leftarrow}(m,n),n)$ for each $m : SL$, $n : TL$.
2) Hippocraticness: if source and target models already satisfy R then the forward and reverse transformations do not modify the models:

$$R(m,n) \Rightarrow R^{\rightarrow}(m,n) = n$$
$$R(m,n) \Rightarrow R^{\leftarrow}(m,n) = m$$

for each $m : SL$, $n : TL$.

The concept of a *lens* is a special case of a bx satisfying these properties [16].

III. BX SPECIFICATION IN UML-RSDS

UML-RSDS is a hybrid model transformation language, with a formal semantics [10] and an established toolset [11]. Model transformations are specified in UML-RSDS as UML use cases, defined declaratively by three main predicates, expressed in a subset of OCL:

1) Assumptions *Asm*, which define when the transformation is applicable.
2) Postconditions *Post*, which define the intended effect of the transformation at its termination. These are an ordered conjunction of OCL constraints (also termed *rules* in the following) and also serve to define a procedural implementation of the transformation.
3) Invariants *Inv*, which define expected invariant properties which should hold during the transformation execution. These may be derived from *Post*, or specified explicitly by the developer.

From a declarative viewpoint, *Post* defines the conditions which should be established by a transformation. From an

 ISBN: 978-1-61208-438-1

implementation perspective, the constraints of *Post* also define intended computation steps of the transformation: each computation step is an application of a postcondition constraint to a specific source model element or to a tuple of elements.

For example, an elementary transformation specification τ_{a2b} on the languages S consisting of entity type A and T consisting of entity type B (Figure 1) could be:

$$(Asm): \quad B{\rightarrow}forAll(b \mid b.y \geq 0)$$
$$(Post): \quad A{\rightarrow}forAll(a \mid B{\rightarrow}exists(b \mid b.y = a.x{\rightarrow}sqr()))$$
$$(Inv): \quad B{\rightarrow}forAll(b \mid A{\rightarrow}exists(a \mid a.x = b.y{\rightarrow}sqrt()))$$

The computation steps α of τ_{a2b} are applications of $B{\rightarrow}exists(b \mid b.y = a.x{\rightarrow}sqr())$ to individual $a : A$. These consist of creation of a new $b : B$ instance and setting its y value to $a.x * a.x$. These steps preserve *Inv*: $Inv \Rightarrow [\alpha]Inv$.

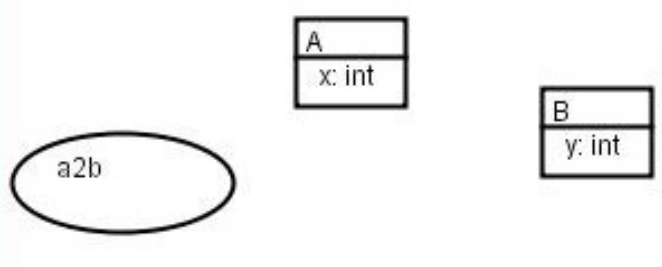


Figure 1. A to B Transformation τ_{a2b}

This example shows a typical situation, where the invariant is a dual to the postcondition, and expresses a form of minimality condition on the target model: that the only elements of this model should be those derived from source elements by the transformation. In terms of the framework of [16], the source-target relation R_τ associated with a UML-RSDS transformation τ is *Post and Inv*. As in the above example, R_τ is not necessarily bijective. The forward direction of τ is normally computed as $stat(Post)$: the UML activity derived from *Post* when interpreted procedurally [10]. However, in order to achieve the correctness and hippocraticness properties, *Inv* must also be considered: before $stat(Post)$ is applied to the source model m, the target model n must be cleared of elements which fail to satisfy *Inv*.

In the *a2b* example, the transformation $\tau^{\times}_{a2b}$ with postcondition constraints:

$$(CleanTarget1): \quad B{\rightarrow}forAll(b \mid not(b.y \geq 0)\ implies\ b{\rightarrow}isDeleted())$$
$$(CleanTarget2): \quad B{\rightarrow}forAll(b \mid not(A{\rightarrow}exists(a \mid a.x = b.y{\rightarrow}sqrt()))\ implies\ b{\rightarrow}isDeleted())$$

is applied before τ_{a2b}, to remove all B elements which fail to be in R_{a2b} with some $a : A$, or which fail to satisfy *Asm*.

This is an example of the Cleanup before Construct pattern (Section IV). Additionally, the $E{\rightarrow}exists(e \mid P)$ quantifier in rule succedents should be procedurally interpreted as "create a new $e : E$ and establish P for e, unless there already exists an $e : E$ satisfying P". That is, the Unique Instantiation pattern [14] should be used to implement 'check before enforce' semantics. The forward transformation $\tau^{\rightarrow}$ is then the sequential composition $\tau^{\times};\ \tau$ of the cleanup transformation and the standard transformation (enhanced by Unique Instantiation).

In the reverse direction, the roles of *Post* and *Inv* are interchanged: elements of the source model which fail to satisfy *Asm*, or to satisfy *Post* with respect to some element of the target model should be deleted:

$$(CleanSource2): \quad A{\rightarrow}forAll(a \mid not(B{\rightarrow}exists(b \mid b.y = a.x{\rightarrow}sqr()))\ implies\ a{\rightarrow}isDeleted())$$

This cleanup transformation is denoted $\tau^{\sim\times}_{a2b}$. It is followed by an application of the normal inverse transformation $\tau^{\sim}_{a2b}$ which has postcondition constraints *Inv* ordered in the corresponding order to *Post*. Again, Unique Instantiation is used for source model element creation. The overall reverse transformation is denoted by $\tau^{\leftarrow}$ and is defined as $\tau^{\sim\times};\ \tau^{\sim}$.

In the case of separate-models transformations with type 1 postconditions (Constraints whose write frame is disjoint from their read frame), *Inv* can be derived automatically from *Post* by syntactic transformation, the *CleanTarget* and *CleanSource* constraints can also be derived from *Post*, and from *Asm*. This is an example of a higher-order transformation (HOT) and is implemented in the UML-RSDS tools.

In general, in the following UML-RSDS examples, τ is a separate-models transformation with source language S and target language T, and postcondition *Post* as an ordered conjunction of constraints of the form:

$$(Cn): \quad S_i{\rightarrow}forAll(s \mid SCond(s)\ implies\ T_j{\rightarrow}exists(t \mid TCond(t)\ and\ P_{i,j}(s,t)))$$

and *Inv* is a conjunction of dual constraints of the form

$$(Cn^{\sim}): \quad T_j{\rightarrow}forAll(t \mid TCond(t)\ implies\ S_i{\rightarrow}exists(s \mid SCond(s)\ and\ P^{\sim}_{i,j}(s,t)))$$

where the predicates $P_{i,j}(s,t)$ define the features of t from those of s, and are invertible: an equivalent form $P^{\sim}_{i,j}(s,t)$ should exist, which expresses the features of s in terms of those of t, and such that

$$S_i{\rightarrow}forAll(s \mid T_i{\rightarrow}forAll(t \mid P_{i,j}(s,t) = P^{\sim}_{i,j}(s,t)))$$

under the assumptions *Asm*. Table I shows some examples of inverses $P^{\sim}$ of predicates P. The computation of these inverses are all implemented in the UML-RSDS tools (the *reverse* option for use cases). More cases are given in [11]. The transformation developer can also specify inverses for particular *Cn* by defining a suitable $Cn^{\sim}$ constraint in *Inv*, for example, to express that a predicate $t.z = s.x + s.y$ should be inverted as $s.x = t.z - s.y$.

Each *CleanTarget* constraint based on *Post* then has the form:

$$(Cn^{\times}): \quad T_j{\rightarrow}forAll(t \mid TCond(t)\ and\ not(S_i{\rightarrow}exists(s \mid SCond(s)\ and\ P_{i,j}(s,t)))\ implies\ t{\rightarrow}isDeleted())$$

Similarly for *CleanSource*.

TABLE I. EXAMPLES OF PREDICATE INVERSES

$P(s,t)$	$P^{\sim}(s,t)$	Condition
$t.g = s.f$	$s.f = t.g$	Assignable features f, g
$t.g = s.f \rightarrow sqrt()$	$s.f = t.g \rightarrow sqr()$	f, g non-negative attributes
$t.g = K * s.f + L$ Numeric constants K, L, $K \neq 0$	$s.f = (t.g - L)/K$	f, g numeric attributes
$t.rr = s.r \rightarrow including(s.p)$	$s.r = t.rr \rightarrow front()$ *and* $s.p = t.rr \rightarrow last()$	rr, r ordered association ends p 1-multiplicity end
$t.rr = s.r \rightarrow append(s.p)$		
$t.rr = s.r \rightarrow sort()$	$s.r = t.rr \rightarrow asSet()$	r set-valued, rr ordered
$t.rr = s.r \rightarrow asSequence()$		
$R(s,t)$ *and* $Q(s,t)$	$R^{\sim}(s,t)$ *and* $Q^{\sim}(s,t)$	
$t.rr = TRef[s.r.idS]$ idS primary key of $SRef$, idT primary key of $TRef$	$s.r = SRef[t.rr.idT]$	rr association end with element type $TRef$, r association end with element type $SRef$
$t.g = s.r.idS$ Attribute g	$s.r = SRef[t.g]$	idS primary key of $SRef$, r association end with element type $SRef$
$T_j[s.idS].rr = TRef[s.r.idSRef]$ r has element type $SRef$, rr has element type $TRef$	$S_i[t.idT].r = SRef[t.rr.idTRef]$	$idS, idSRef$ primary keys of S_i, $SRef$ $idT, idTRef$ primary keys of T_j, $TRef$

IV. PATTERNS FOR BX

In this section, we give a patterns catalogue for bx, and give pattern examples in UML-RSDS.

A. Auxiliary Correspondence Model

This pattern defines auxiliary entity types and associations which link corresponding source and target elements. These are used to record the mappings performed by a bx, and to propagate modifications from source to related target elements or vice-versa, when one model changes.

Figure 2 shows a typical schematic structure of the pattern.

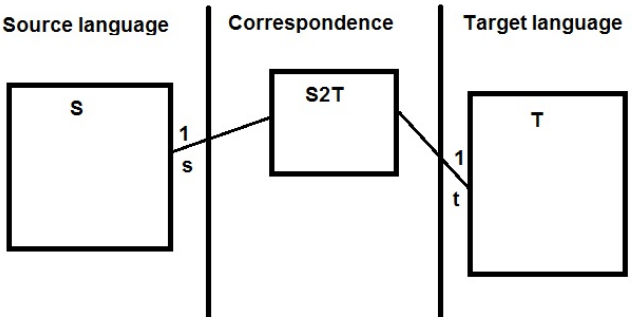


Figure 2. Auxiliary Correspondence Model pattern

Benefits: The pattern is a significant aid in change-propagation between models, and helps to ensure the correctness of a bx. Feature value changes to a source element s can be propagated to changes to its corresponding target element, and vice-versa, via the links. Deletion of an element may imply deletion of its corresponding element.

Disadvantages: The correspondence metamodel must be maintained (by the transformation engineer) together with the source and target languages, and the necessary actions in creating and accessing correspondence elements adds complexity to the transformation and adds to its execution time and memory requirements.

Related Patterns: This pattern is a specialisation of the *Auxiliary Metamodel* pattern of [14].

Examples: This mechanism is a key facility of Triple Graph Grammars (TGG) [1][2], and correspondence traces are maintained explicitly or implicitly by other MT languages such as QVT-R [15].

In UML-RSDS, the pattern is applied by introducing auxiliary attributes into source and target language entity types. These attributes are primary key/identity attributes for the entity types, and are used to record source-target element correspondences. Target element $t : T_j$ is considered to correspond to source element(s) $s_1 : S_1$, ..., $s_n : S_n$ if they all have the same primary key values: $t.idT_j = s_1.idS_1$, etc. The identity attributes are String-valued in this paper. The correspondence between a source entity S_i and a target entity T_j induced by equality of identity attribute values defines a *language mapping* or *interpretation* χ of S_i by T_j in the sense of [13]:

$$S_i \longmapsto T_j$$

with $S_i \rightarrow collect(idS_i) = T_j \rightarrow collect(idT_j)$.

The existence of identity attributes facilitates element lookup by using the *Object Indexing* pattern [14], which defines maps from *String* to each entity type, permitting elements to be retrieved by the value of their identity attribute: $T_j[v]$ denotes the T_j instance t with $t.idT_j = v$ if v is a single String value, or the collection of T_j instances t with $v \rightarrow includes(t.idT_j)$ if v is a collection. The last three cases in Table I show inverse predicates derived using this approach to correspondence models. Note that $T_j[x.idT_j] = x$ for $x : T_j$.

The pattern can be used to define source-target propagation and incremental application of a transformation τ. For

postconditions Cn of the form

$$S_i \rightarrow forAll(s \mid SCond(s)\ implies\ T_j \rightarrow exists(t \mid TCond(t)\ and\ P_{i,j}(s,t)))$$

derived constraints Cn^{Δ} can be defined for the incremental application of Cn to model increments (finite collections of creations, deletions and modifications of model elements).

The incremental version τ^{Δ} of a transformation τ is defined to have postconditions formed from the constraints Cn^{Δ} for each postcondition Cn of τ, and ordered according to the order of the Cn in the $Post$ of τ. In a similar way, target-source change propagation can be defined. Change propagation is implemented in UML-RSDS by the *incremental* mode of use case execution.

B. Cleanup before Construct

This pattern defines a two-phase approach in both forward and reverse transformations associated with a bx with relation R: the forward transformation $R^{\rightarrow}$ first removes all elements from the target model n which fail to satisfy R for any element of the source m, and then constructs elements of n to satisfy R with respect to m. The reverse transformation $R^{\leftarrow}$ operates on m in the same manner.

Benefits: The pattern is an effective way to ensure the correctness of separate-models bx.

Disadvantages: There may be efficiency problems because for each target model element, a search through the source model for possibly corresponding source element may be needed. Elements may be deleted in the Cleanup phase only to be reconstructed in the Construct phase. Auxiliary Correspondence Model may be an alternative strategy to avoid this problem, by enforcing that feature values should change in response to a feature value change in a corresponding element, rather than deletion of elements.

Related Patterns: This pattern is a variant of the *Construction and Cleanup* pattern of [14].

Examples: An example is the Composers bx [4]. Implicit deletion in QVT operates in a similar manner to this pattern, but can only modify models (domains) marked as *enforced* [15]. In UML-RSDS, explicit cleanup rules $Cn^{\times}$ can be deduced from the construction rules Cn, for mapping transformations, as described in Section III above. If identity attributes are used to define the source-target correspondence, then $Cn^{\times}$ can be simplified to:

$$T_j \rightarrow forAll(t \mid TCond(t)\ and\ S_i \rightarrow collect(sId) \rightarrow excludes(t.tId)\ implies\ t \rightarrow isDeleted())$$

and

$$T_j \rightarrow forAll(t \mid TCond(t)\ and\ S_i \rightarrow collect(sId) \rightarrow includes(t.tId)\ and\ s = S_i[t.tId]\ and\ not(SCond(s))\ implies\ t \rightarrow isDeleted())$$

In the case that $TCond(t)$ and $SCond(s)$ hold for corresponding s, t, but $P_{i,j}(s,t)$ does not hold, t should not be deleted, but $P_{i,j}(s,t)$ should be established by updating t:

$$S_i \rightarrow forAll(s \mid T_j \rightarrow collect(tId) \rightarrow includes(s.sId)\ and\ t = T_j[sId]\ and\ SCond(s)\ and\ TCond(t)\ implies\ P_{i,j}(s,t))$$

For a transformation τ, the cleanup transformation $\tau^{\times}$ has the above $Cn^{\times}$ constraints as its postconditions, in the same order as the Cn occur in the $Post$ of τ. Note that $\tau^{\rightarrow}$ is $\tau^{\times}$; τ, and τ^{Δ} is $\tau^{\times}$; τ incrementally applied.

C. Unique Instantiation

This pattern avoids the creation of unnecessary elements of models and helps to resolve possible choices in reverse mappings. It uses various techniques such as traces and unique keys to identify when elements should be modified and reused instead of being created. In particular, unique keys can be used to simplify checking for existing elements.

Benefits: The pattern helps to ensure the Hippocraticness property of a bx by avoiding changes to a target model if it is already in the transformation relation with the source model. It implements the principle of 'least change' [17].

Disadvantages: The need to test for existence of elements adds to the execution cost. This can be ameliorated by the use of the Object Indexing pattern [14] to provide fast lookup of elements by their primary key value.

Examples: The *key* attributes and check-before-enforce semantics of QVT-R follow this pattern, whereby new elements of source or target models are not created if there are already elements, which satisfy the specified relations of the transformation [16]. The $E \rightarrow exists1(e \mid P)$ quantifier in UML-RSDS is used in a similar way. It is procedurally interpreted as "create a new $e : E$ and establish P for e, unless there already exists an $e : E$ satisfying P" [11]. For bx, the quantifier *exists* should also be treated in this way. If a transformation uses identity attributes (to implement Auxiliary Correspondence Model), the quantifier $E \rightarrow exists(e \mid e.eId = v\ and\ P)$ can be interpreted as: "if $E[v]$ exists, apply $stat(P)$ to this element, otherwise create a new E instance with $eId = v$ and apply $stat(P)$ to it". This ensures Hippocraticness.

D. Phased Construction for bx

This pattern defines a bx τ by organising R_{τ} as a union of relations $R_{Si,Tj}$ which relate elements of entities Si and Tj which are in corresponding levels of the composition hierarchies of the source and target languages. Figure 3 shows the typical schematic structure of the pattern. At each composition level there is a 0..1 to 0..1 relation (or more specialised relation) between the corresponding source and target entity types.

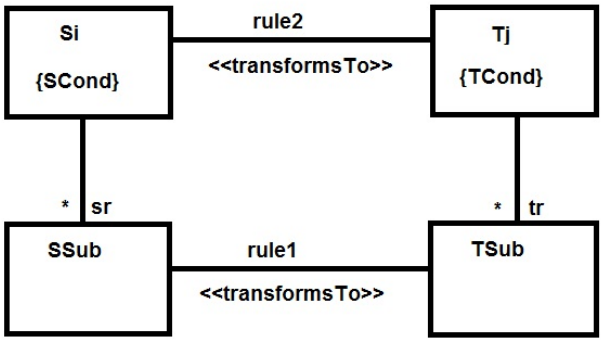


Figure 3. Phased Construction pattern

Benefits: The pattern provides a modular and extensible means to structure a bx.

Examples: The UML to relational database example of [15] is a typical case, where *Package* and *Schema* correspond at the top of the source/target language hierarchies, as do *Class* and *Table* (in the absence of inheritance), and *Column* and *Attribute* at the lowest level.

In UML-RSDS a transformation defined according to this pattern has its *Post* consisting of constraints *Cn* of the form

$$S_i{\rightarrow}forAll(s \mid SCond(s)\ implies \\ T_j{\rightarrow}exists(t \mid TCond(t)\ and\ P_{i,j}(s,t)))$$

where S_i and T_j are at corresponding hierarchy levels, and *Inv* consists of constraints $Cn^{\sim}$ of the form

$$T_j{\rightarrow}forAll(t \mid TCond(t)\ implies \\ S_i{\rightarrow}exists(s \mid SCond(s)\ and\ P^{\sim}_{i,j}(s,t)))$$

No nested quantifiers or deletion expressions $x{\rightarrow}isDeleted()$ are permitted in *SCond*, *TCond* or $P_{i,j}$, and $P_{i,j}$ is restricted to be formed of invertible expressions.

Each rule creates elements t of some target entity type T_j, and may lookup target elements produced by preceding rules to define the values of association end features of t: $t.tr = TSub[s.sr.idSSub]$ for example, where *TSub* is lower than T_j in the target language composition hierarchy (as in Figure 3) and there are identity attributes in the entities to implement a source-target correspondence at each level. Both forward and reverse transformations will conform to the pattern if one direction does. The assignment to $t.tr$ has inverse: $s.sr = SSub[t.tr.idTSub]$.

Two UML-RSDS bx $\tau : S \rightarrow T$, $\sigma : T \rightarrow U$ using this pattern can be sequentially composed to form another bx between S and U: the language T becomes auxiliary in this new transformation. The forward direction of the composed transformation is $\tau^{\rightarrow}; \sigma^{\rightarrow}$, the reverse direction is $\sigma^{\leftarrow}; \tau^{\leftarrow}$.

E. Entity Merging/Splitting for bx

In this variation of Phased Construction, data from multiple source model elements may be combined into single target model elements, or vice-versa, so that there is a many-one relation from one model to the other. The pattern supports the definition of such bx by including correspondence links between the multiple elements in one model which are related to one element in the other.

Benefits: The additional links enable the transformation to be correctly reversed.

Disadvantages: Additional auxiliary data needs to be added to record the links. The validity of the links between elements needs to be maintained. There may be potential conflict between different rules which update the same element.

Related Patterns: This uses a variant of Auxiliary Correspondence Model, in which the correspondence is between elements in one model, in addition to cross-model correspondences. The attributes used to record intra-model correspondences may not be primary keys.

Examples: An example of Entity Merging is the Collapse/Expand State Diagrams benchmark of [6]. The UML to RDB transformation is also an example in the case that all subclasses of a given root class are mapped to a single table that represents this class. The Pivot/Unpivot transformation of [3] is an example of Entity Splitting.

In the general case of merging/splitting, the inverse of C_n:

$$S_{i1}{\rightarrow}forAll(s1 \mid ... \\ S_{in}{\rightarrow}forAll(sn \mid SCond(s1,...,sn)\ implies \\ T_{j1}{\rightarrow}exists(t1 \mid ... \\ T_{jm}{\rightarrow}exists(tm \mid TCond(t1,...,tm)\ and \\ P(s1,...,sn,t1,...,tm))...))\ ...)$$

is $Cn^{\sim}$:

$$T_{j1}{\rightarrow}forAll(t1 \mid ... \\ T_{jm}{\rightarrow}forAll(tm \mid TCond(t1,...,tm)\ implies \\ S_{i1}{\rightarrow}exists(s1 \mid ... \\ S_{in}{\rightarrow}exists(sn \mid SCond(s1,...,sn)\ and \\ P^{\sim}(s1,...,sn,t1,...,tm))...))...)$$

In UML-RSDS, correspondence links between elements in the same model are maintained using additional attributes. All elements corresponding to a single element will have the same value for the auxiliary attribute (or a value derived by a 1-1 function from that value).

F. Map Objects Before Links for bx

If there are self-associations on source entity types, or other circular dependency structures in the source model, then this variation on Phased Construction for bx can be used. This pattern separates the relation between elements in target and source models from the relation between links in the models.

Benefits: The specification is made more modular and extensible. For example, if a new association is added to one language, and a corresponding association to the other language, then a new relation relating the values of these features can be added to the transformation without affecting the existing relations.

Disadvantages: Some features of one entity type are treated in separate relations.

Examples: In UML-RSDS a first phase of such a transformation relates source elements to target elements, then in a second phase source links are related to corresponding target links. The second phase typically has postcondition constraints of the form $S_i{\rightarrow}forAll(s \mid T_j[s.idS].rr = TRef[s.r.idSRef])$ to define target model association ends *rr* from source model association ends *r*, looking-up target model elements $T_j[s.idS]$ and $TRef[s.r.idSRef]$ which have already been created in a first phase. Such constraints can be inverted to define source data from target data as: $T_j{\rightarrow}forAll(t \mid S_i[t.idT].r = SRef[t.rr.idTRef])$. The reverse transformation also conforms to the Map Objects Before Links pattern.

An example of this pattern is the tree to graph transformation [9], Figure 4.

A first rule creates a node for each tree:

$$Tree{\rightarrow}forAll(t \mid Node{\rightarrow}exists(n \mid n.label = t.label))$$

A second rule then creates edges for each link between parent and child trees:

$$Tree{\rightarrow}forAll(t \mid \\ Tree{\rightarrow}forAll(p \mid t.parent{\rightarrow}includes(p)\ implies \\ Edge{\rightarrow}exists(e \mid e.source = Node[t.label]\ and \\ e.target = Node[p.label])))$$

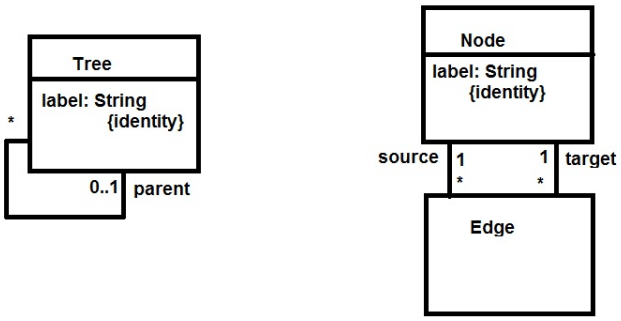


Figure 4. Tree to graph metamodels

The corresponding invariant predicates, defining the reverse transformation, are:

$$Node{\rightarrow}forAll(n \mid Tree{\rightarrow}exists(t \mid t.label = n.label))$$

and

$$\begin{array}{l} Edge{\rightarrow}forAll(e \mid \\ \quad Tree{\rightarrow}exists(t \mid Tree{\rightarrow}exists(p \mid \\ \qquad t.parent{\rightarrow}includes(p) \ and \\ \qquad t.label = e.source.label \ and \\ \qquad p.label = e.target.label))) \end{array}$$

Inv is derived mechanically from *Post* using Table I, and provides an implementable reverse transformation, since $stat(Inv)$ is defined.

V. Related Work

There are a wide range of approaches to bx [8]. Currently the most advanced approaches [2][5] use constraint-based programming techniques to interpret relations $P(s,t)$ between source and target elements as specifications in both forward and reverse directions. These techniques would be a potentially useful extension to the syntactic inverses defined in Table I, however the efficiency of constraint programming will generally be lower than the statically-computed inverses. The approach also requires the use of additional operators extending standard OCL. Further techniques include the inversion of recursively-defined functions [18], which would also be useful to incorporate into the UML-RSDS approach.

In [13] we identify the role of *language interpretations* $\chi : S \rightarrow T$ in specifying transformations. Interpretations are closely related to transformation inversion: at the model level a mapping $Mod(\chi) : Mod(T) \rightarrow Mod(S)$ from the set of models of T to those of S can be defined based on χ: the interpretation of an S language element E in $Mod(\chi)(n)$ for $n : Mod(T)$ is that of $\chi(E)$ in n. Syntactically, the inverse of a transformation τ specified by a language morphism χ can be derived from χ: the value of a feature f of source element s of source entity E is set by $s.f = t.\chi(E :: f)$ in the case of an attribute, and by $s.r = SRef[t.\chi(E :: r).idTRef]$ in the case of a role with element type $SRef$.

Considerable research has been carried out on the theory of bx. One principle which has been formulated for bx is the *principle of least change* [17]. This means that a bx which needs to modify one model in order to re-establish the bx relation R with a changed other model, should make a minimal possible such change to the model. In our approach, *Post* in the forward direction, and *Inv* in the reverse direction, express the necessary minimal conditions for the models to be related by R. The synthesised implementation of these constraints as executable code carries out the minimal changes necessary to establish *Post* and *Inv*, and hence satisfies the principle of least change.

VI. Conclusion

We have defined a declarative approach for bidirectional transformations based on the derivation of forward and reverse transformations from a specification of dual postcondition and invariant relations between source and target models. The approach enables a wide range of bx to be defined, including cases of many-to-one and one-to-many relations between models, in addition to bijections. We have described transformation patterns which may be used to structure bx. The derivation of reverse transformations has been implemented in the UML-RSDS tools [11].

References

[1] A. Anjorin and A. Rensink, “SDF to Sense transformation”, TU Darmstadt, Germany, 2014.

[2] A. Anjorin, G. Varro, and A. Schurr, “Complex attribute manipulation in TGGs with constraint-based programming techniques”, BX 2012, Electronic Communications of the EASST vol. 49, 2012.

[3] M. Beine, N. Hames, J. Weber, and A. Cleve, “Bidirectional transformations in database evolution: a case study ‘at scale’”, EDBT/ICDT 2014, CEUR-WS.org, 2014.

[4] J. Cheney, J. McKinna, P. Stevens, and J. Gibbons, “Towards a repository of bx examples”, EDBT/ICDT 2014, 2014, pp. 87–91.

[5] A. Cicchetti, D. Di Ruscio, R. Eramo, and A. Pierantonio, “JTL: a bidirectional and change propagating transformation language”, SLE 2010, LNCS vol. 6563, 2011, pp. 183–202.

[6] K. Czarnecki, J. Nathan Foster, Z. Hu, R. Lammel, A. Schurr, and J. Terwilliger, “Bidirectional transformations: a cross-discipline perspective”, GRACE workshop, ICMT, 2009.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1994.

[8] Z. Hu, A. Schurr, P. Stevens, and J. Terwilliger (eds.), “Report from Dagstuhl Seminar 11031”, January 2011, www.dagstuhl.de/11031.

[9] D. S. Kolovos, R. F. Paige, and F. Polack, “The Epsilon Transformation Language”, ICMT, 2008, pp. 46–60.

[10] K. Lano and S. Kolahdouz-Rahimi, “Constraint-based specification of model transformations”, Journal of Systems and Software, vol. 88, no. 2, February 2013, pp. 412–436.

[11] K. Lano, The UML-RSDS Manual, www.dcs.kcl.ac.uk/staff/kcl/uml2web/umlrsds.pdf, 2015.

[12] K. Lano, S. Kolahdouz-Rahimi, and K. Maroukian, “Solving the Petri-Nets to Statecharts Transformation Case with UML-RSDS”, TTC 2013, EPTCS, 2013, pp. 101–105.

[13] K. Lano and S. Kolahdouz-Rahimi, “Towards more abstract specification of model transformations”, ICTT 2014.

[14] K. Lano and S. Kolahdouz-Rahimi, “Model-transformation Design Patterns”, IEEE Transactions in Software Engineering, vol 40, 2014, pp. 1224–1259.

[15] OMG, MOF 2.0 Query/View/Transformation Specification v1.1, 2011.

[16] P. Stevens, “Bidirectional model transformations in QVT: semantic issues and open questions”, SoSyM, vol. 9, no. 1, January 2010, pp. 7–20.

[17] Theory of Least Change, groups.inf.ed.ac.uk/bx/, accessed 3.9.2015.

[18] J. Voigtlander, Z. Hu, K. Matsuda, and M. Wang, “Combining syntactic and semantic bidirectionalization”, ICFP ’10, ACM Press, 2010, pp. 181–192.

 ISBN: 978-1-61208-438-1

Towards a Framework for Software Product Maturity Measurement

Mohammad Alshayeb[1], Ahmad Khader Abdellatif[1], Sami Zahran[2] and Mahmood Niazi[1]
[1]: Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran, 31261 Saudi Arabia
e-mail: {alshayeb, aabdellatif, mkniazi}@kfupm.edu.sa

[2]: Intelligent Consultancy & Training (ICT) Limited, United Kingdom
e-mail: Sami_zahran@hotmail.com

***Abstract*—Capability Maturity Model Integration (CMMI) is a software process improvement model that aims at improving the processes of the software development. CMMI focuses on the "process quality" instead of "product quality". Studies have shown that focusing on "process quality" alone does not guarantee the quality of the produced software, whereas equal attention to product quality is also essential for ensuring the overall software quality. The objective of this paper is to present the initial structure of the framework we propose to measure and assess the software product maturity level. The measure we use for the product maturity is the level of the product compliance with the internal and external quality attributes defined in the stakeholders' requirements. In this framework, we focus on the quality of the product of the process. The proposed framework will help assess the quality of the software product through assessment of the final software deliverable. Successful implementation of the proposed framework will provide a better insight of the software product quality, hence its maturity. We refer to any deliverable code as a product.**

Keywords-Software Product Quality; Software Product Maturity; Product Maturity Assessment; Product Maturity Levels; Product Maturity Model Integration (PMMI); Product Maturity Assessment Method (PMAM).

I. INTRODUCTION

The Software Engineering Institute (SEI) of Carnegie Mellon University (CMU) defines the Capability Maturity Model Integration (CMMI) as a process improvement approach that provides organizations with the essential elements of effective processes to improve their software development performance. CMMI process improvement includes identifying the organization's process strengths and weaknesses and making process changes to convert weaknesses into strengths [1]. CMMI consists of best practices that help organizations to improve their software development effectiveness, efficiency, and quality [2].

CMMI defines three constellations, which are collections of best practices and process improvement goals that organizations use to evaluate and improve their processes. These goals and practices are organized into different process areas. The three constellations are:

1. The CMMI for Acquisition (CMMI-ACQ): provides guidance to organizations that manage the supply chain to acquire products and services that meet the needs of the customer.
2. The CMMI for Development (CMMI-DEV): provides process improvement guidance to organizations that develop products and services.
3. The CMMI for Services (CMMI-SVC): provides guidance to organizations that establish, manage, and deliver services that meet the needs of customers and end users.

CMMI aims at improving the process of the software development, however, that does not guarantee the quality of the produced software as the focus in CMMI does not cover "product quality". Previous research have shown that dealing with only "process quality" is not sufficient and that assessment of "product quality" is also required for the improvement of overall software quality [3]. Our proposed framework described in this paper focuses on the quality of the product instead of the process. The quality/maturity of the software product can be assured through the assessment of deliverables of the major phases of the software development lifecycle. Our proposed framework adopts a method for technical product evaluation and quality assessment as the basis for establishing the product's level of maturity. The level of product maturity measured by the degree of its compliance with the internal and external quality attributes defined in the stakeholders requirements. We call this framework Technical-CMMI (T-CMMI). The proposed framework along with the assessment method will: 1) enable software companies to assess their software products to ensure they meet the desired quality before they release it to their clients, 2) enable clients to evaluate the quality of the product before purchasing it and 3) provide the clients with the ability to compare between the quality of different software products.

The rest of this paper is organized as follows: we present the related work in Section 2. In Section 3, we describe the proposed framework. Finally, in Section 4, we present the conclusions and future work.

 ISBN: 978-1-61208-438-1

II. Related Work

In this section, we review the literature on developing maturity models and in software certification for quality assessment.

A. Software Product Maturity Models

Researchers proposed different product maturity models. Al-Qutaish et al. [4] proposed a software product quality maturity model (SPQMM) for assessing the quality of the software product. The proposed model is based on ISO 9126, Six Sigma, and ISO 15026. The model uses the characteristics, sub-characteristics, and measurements of ISO 9126. The values are combined into a single value, which are converted to six sigma. After that, the integrity level of the software product using ISO 15026 is calculated. Finally, the maturity level of the software product is identified. SPQMM is limited to the quality attributes and metrics defined in ISO/IEC 9126 standard.

The EuroScope consortium [5] proposed SCOPE Maturity Model (SMM), a maturity model of software products evaluation. The model has five maturity levels: initial, repeatable, defined, managed, and optimizing. SMM levels 2, 3, and 4 use ISO 12119, ISO/IEC 9126, and ISO 14598 standards. SMM is a measure of the quality in terms of matching stated specifications or requirements; tests are executed to assess the degree to which a product meets the required specifications. SMM requires the process to be documented to ensure the product matches the specifications. Thus, SMM does not focus on the final product quality (code).

April et al. [6] proposed the Software Maintenance Maturity Model (SMmm) However, SMmm focuses only on maintainability. Alvaro et al. [7] proposed a Software Component Maturity Model (SCMM) that is based on ISO/IEC9126 and ISO/IEC 14598 standards. SCMM contains five levels. SCMM depends mainly on the component quality model (CQM). SCMM measures only the maturity of the components and it cannot assess different types of product such as enterprise applications, web-services. Golden et al. [8] proposed the Open Source Maturity Model (OSMM) which helps in assessing and comparing open source software products to identify which one is the best for a defined application. OSMM evaluates the maturity of open source products only without assessing the quality of these software products. OSMM is not primarily used to assess software product quality attributes or product maturity but to help organizations perform a comparison between open source systems.

These three models above either

- Use limited set of quality attributes [4], do not focus on measuring the final software quality [5], or
- Have limited scope [6]-[8].

Therefore, the proposed model will overcome all these limitations.

Our proposed model is designed to be flexible to enable the assessor(s) to define their own set of quality attributes and metrics (based on the stakeholders requirements). In addition, it is generic enough to be applicable to any type of software domain, size or development method.

B. Software Product Certifications

Our proposed model can also serve in certifying software products. Software certification can be granted for different types of software such as final software products [9-13] and components [14]. Certification can be provided by independent agencies, which function like other quality agencies. Involving external agencies in providing the certificate increases the trust in the certification as indicated by Voas [15] “completely independent product certification offers the only approach that consumers can trust”. Most of the certification methods are process-based [16], from the process they can determine the quality of the final product. However, certifying the software development process only does not guarantee the quality of the final product [3].

III. Framework for Software Product Maturity Model Integration

In this section, we describe the proposed product maturity assessment framework that can be used to assess the maturity of software products. T-CMMI follows the CMMI approach in defining a reference model and assessment method. T-CMMI consists of two parts:

1. Reference Model that describes the common basis for the assessors to assess the maturity of software products. The reference model describes a scale of the maturity/capability levels of the software product based on its degree of compliance with a set of quality attributes and metrics defined in the stakeholders’ requirements.
2. Assessment Method that describes how to use the reference model in assessing the final software product. It also provides guidelines and checklists that help in the assessment process and to ensure a common base of judgment.

Both reference model and the assessment method of the T-CMMI are shown in Figure 1.

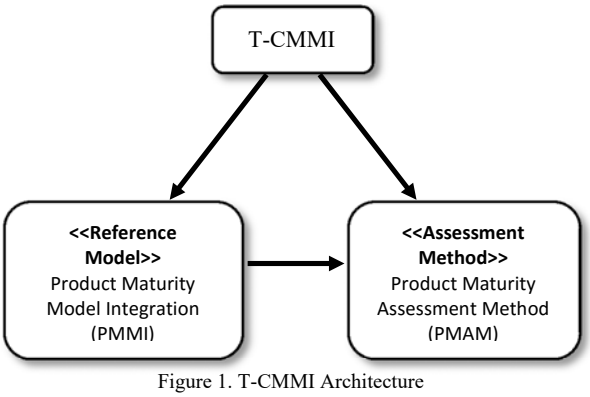


Figure 1. T-CMMI Architecture

We adopted CMMI structure for the development of T-CMMI architecture, which contains a reference model and an assessment method. The reference model for the T-CMMI is

called Product Maturity Model Integration (PMMI), which contains the capability and product maturity levels. PMMI contains a predefined set of quality attributes and metrics to measure these quality attributes. PMMI adopts the focus-area maturity model structure as opposed to the fixed levels maturity model structure adopted by CMMI. PMMI has two focus-areas, which concentrate on the internal and external quality attributes of the product. The purpose of the Reference Model is to provide a platform and a focus for gathering evidence for product quality indicators that will be used to assess the product maturity level during the Product Maturity Assessment.

The assessment method is called Product Maturity Assessment Method (PMAM). PMAM defines the steps for assessing the final software product against the reference model maturity levels. PMAM contains guidelines and checklists to illustrate how the assessors follow the guidelines in order to measure the capability level and product maturity level for both of PMMI's focus-areas, which concentrate on the internal and external quality attributes. The purpose of the PMAM is to provide a standard method for assessing the product maturity/capability by assessing the degree to which the product conforms to the stakeholders required quality attributes. Below, we discuss these two components in details.

A. Product Maturity Model Integration (PMMI)

PMMI defines a reference model for assessing product maturity and capability. The scope of the PMMI reference model covers integrated view to the end-to-end lifecycle starting with product requirements and ending with product integration, testing and release. The lifecycle is divided into two stages, the DEV stage and the REL stage. These two stages are separate Functional Domains (containing all activities and actors that are involved in the set of activities defined in the development methodology being followed). Each of the DEV & REL stage will have its own Set of Stakeholders and product quality attributes. These two functional domains are defined as follows:

- The DEV stage: covers all the processes and activities for software development, integration and testing (both software unit and software integration testing) of the product. The outcome of the DEV stage is a product ready to be transitioned to the REL stage.
- The REL stage: covers system integration and product pre-release testing

Figure 2 illustrates the PMMI structure showing the DEV and REL stages. Figure 2 shows the main components of each PMMI stage. On the left side are DEV-Stage components, which focus on measuring internal quality attributes, while on the right side are REL-Stage components, which focus on external quality attributes. Product maturity assessment component contains the metrics for each quality attribute that are measured and their results are collected to calculate the capability level for each quality attribute. Then, the capability level of all quality attributes will be fetched into PMMI internal/external quality attributes components. In PMMI internal/external quality attributes component, the weighted average capability values of all quality attributes is calculated to measure the stage maturity level. Finally, the calculated maturity level will be the input to Aggregated DEV/REL Stage Maturity Level component where it is rounded down to calculate the stage maturity level.

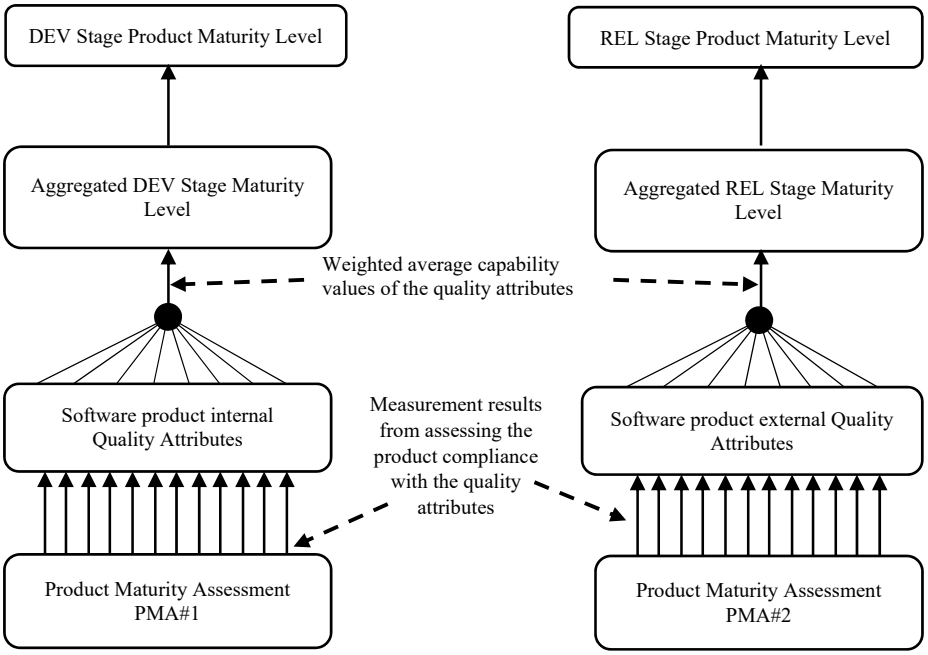


Figure 2. Components of the Product Maturity Model Integration (PMMI)

B. Product Maturity Assessment Method (PMAM)

The PMAM assessment method covers the activities necessary to determine the extent of a product capability to perform in a full compliance with stakeholders' quality requirements. The scope of the assessment is to assess a software product's degree of compliance with the quality attributes defined by the stakeholders (agreed in advance with the assessment sponsor) that covers an integrated view of the end-to-end lifecycle starting with the product and ending with product integration, testing and release. The purpose of the PMAM is to provide a standard method for assessing the level of the product maturity/capability by assessing the degree of the product's conformance with the stakeholders required quality attributes. The PMAM method is compliant with "Guidance on Performing an Assessment" ISO model (ISO 15504-3) [17] framework for software assessment in specifying and defining:

1. Assessment Input.
2. Assessment Process.
3. Assessment Output.
4. Identity of assessment sponsors
5. Identity of Assessors.
6. Responsibilities of each PMAM team member.
7. Expected assessment output and minimum data that should be included in the final assessment report

C. T-CMMI Flexibility

Both components of T-CMMI (PMMI and PMAM) are designed to be flexible and independent of the specific development methodology. In PMMI, assessors can 1) define the quality attributes of interest to the relevant stakeholders with no limits as ISO 9126 defines six attributes only, 2) select the metrics used to measure these quality attributes and 3) define the target capability and maturity levels and their threshold.

PMAM is also designed to be flexible. PMAM process, 1) is applicable to all software domains, 2) can be applied to all software with any size and complexity, and 3) is applicable to all software development lifecycles regardless of the process (or the development methodology) used to build it.

IV. Conclusion and Future Work

This paper presented an approach towards developing a software product maturity model. The proposed framework gives the ability to measure the maturity of a software product of any size and domain. It is also applicable to all software regardless of the process used to build it. T-CMMI framework is designed to be flexible, however, assessors can always use the pre-defined set of quality attributes and metrics (which will be supplied with the model) if they wish without customization.

T-CMMI will complement CMMI as CMMI assesses the process quality while T-CMMI assesses product quality. We expect that companies with higher CMMI level should produce better products measured by T-CMMI framework.

In our future work, we plan to complete the development and evaluation of the framework. We will also develop a website to automate the assessment process.

Acknowledgement

The authors would like to acknowledge the support provided by King Abdul-Aziz City for Science and Technology (KACST) through the Science & Technology Unit at King Fahd University of Petroleum & Minerals (KFUPM) for funding this work through project no. 12-INF3012-04 as part of the National Science, Technology and Innovation Plan.

References

[1] SEI-CMU. *Capability Maturity Model Integration*. Accessed (August 2015), Available: http://www.sei.cmu.edu/cmmi/

[2] SEI-CMU. *CMMI Process Areas*. Accessed (August 2015), Available: http://cmmiinstitute.com/cmmi-overview/cmmi-process-areas/

[3] T. Maibaum and A. Wassyng, "A product-focused approach to software certification," *Computer,* vol. 41, 2008, pp. 91-93.

[4] R. Qutaish and A. Abran, "A maturity model of software product quality," *Journal of Research and Practice in Information Technology,* vol. 43, 2011, pp. 307-327.

[5] A. B. Jakobsen, M. O'Duffy, and T. Punter, "Towards a maturity model for software product evaluations," in *Proceedings of 10th european conference on software cost estimation (ESCOM'99)*, 1999.

[6] A. April, J. Huffman Hayes, A. Abran, and R. Dumke, "Software Maintenance Maturity Model (SMmm): the software maintenance process model," *Journal of Software Maintenance and Evolution: Research and Practice,* vol. 17, 2005, pp. 197-223.

[7] A. Alvaro, E. S. de Almeida, and S. L. Meira, "A Software Component Maturity Model (SCMM)," in *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on*, 2007, pp. 83-92.

[8] B. Golden, *Succeeding with open source*: Addison-Wesley Professional, 2005.

[9] R. Baggen, J. P. Correia, K. Schill, and J. Visser, "Standardized code quality benchmarking for improving software maintainability," *Software Quality Journal,* vol. 20, 2012, pp. 287-307.

[10] J. P. Correia and J. Visser, "Certification of technical quality of software products," in *Proc. of the Int'l Workshop on Foundations and Techniques for Open Source Software Certification*, 2008, pp. 35-51.

[11] P. Heck, M. Klabbers, and M. van Eekelen, "A software product certification model," *Software Quality Journal,* vol. 18, 2010/03/01 2010, pp. 37-55.

[12] P. M. Heck, "A Software Product Certification Model for Dependable Systems " Eindhoven: Technische Universiteit Eindhoven 2006.

[13] J. H. Yahaya, A. Deraman, and A. R. Hamdan, "SCfM_PROD: A software product certification model," in *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, 2008, pp. 1-6.

 ISBN: 978-1-61208-438-1

[14] A. Alvaro, E. S. d. Almeida, and S. L. Meira, "Towards a Software Component Certification Framework," in *Proceedings of the Seventh International Conference on Quality Software*, 2007, pp. 298-303.
[15] J. Voas, "Developing a usage-based software certification process," *Computer*, vol. 33, 2000, pp. 32-37.
[16] J. Morris, G. Lee, K. Parker, G. A. Bundell, and C. P. Lam, "Software component certification," *Computer*, vol. 34, 2001, pp. 30-36.
[17] ISO/IEC, "15504-3: Information Technology - Process Assessment - Part 3 - Guidance on Performing an Assessment No. 15504-3," 2004.

 ISBN: 978-1-61208-438-1

An Exploratory Study on the Influence of Developers in Code Smell Introduction

Leandro Alves, Ricardo Choren, Eduardo Alves
Military Institute of Engineering - IME
Computer Science's Departament
RJ, Brazil
Email: leansousa@gmail.com, choren@ime.eb.br, eduaopec@yahoo.com.br

***Abstract*—A code smell is any symptom in the source code that possibly indicates a deeper maintainability problem. Code smell introduction is a creative task - developers unintentionally introduce code smells in their programs. In this study, we try to obtain a deeper understanding on the relationship between developers and code smell introduction on a software. We analyzed instances of code smells previously reported in the literature and our study involved over 6000 commits of 5 open source object-oriented systems. First, we analyzed the distributions of developers using specific characteristics to classify the developers into groups. Then, we investigated the relationships between types of developers and code smells. The outcome of our evaluation suggests that the way a developer participates in the project may be associated with code smell introduction.**

Keywords–Code smells; exploratory study; software development and maintenance; development teams

I. INTRODUCTION

Software development is a complex activity that does not end even when the software is delivered. Usually, a software needs to be modified to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment [1]. However, continuous change can degrade the system maintainability. The degree of maintainability of a software system can be defined as the degree of ease that the software can be understood, adjusted, adapted, and evolved, and comprises aspects that influence the effort required to implement changes, perform modifications and removal of defects [2]. There are several issues that decrease the maintainability of a software system, such as problems with design principles, lack of traceability between analysis and design documentation, source code without comments and code smells.

Code smells are characteristics of the software that may indicate a code or design problem that can make software hard to evolve and maintain [3]. For instance, the more parameters a method has, the more complex it is. It would be desirable to limit the number of parameters you need in a given method, or use an object to combine the parameters. The presence of code smells indicates that there are issues with code quality, such as understandability and changeability, which can lead to maintainability problems [4].

The code quality depends on how good the developers are. However, there is little knowledge about the influence of developers on the introduction of code smells in a software system. Previous work focus on code smell detection and removal [5][6] and other studies focus on the awareness about code smells on the developer's side [4][6]. The challenge is to further understand the relationship between developers and code smell introduction. As a result, software managers have little knowledge on how the development team affects the software maintainability.

There are still some questions regarding the interplay between developers and the existence of code smells in a source code. Can the way how a developer Works in a Project, be used to understand the frequency of some code smell introduction in a source code? What types of code smells a developer is more likely to introduce? Understanding these issues may help developers to improve their skills and to build team culture with the purpose of avoiding code smells.

This paper presents a study to assess the influence of developers in code smell introduction in software code. Our investigation focused on the study of five software maintenance projects. The projects were selected because of the following characteristics: they were open source projects; information about them were available in a Git repository [7]; they had a substantial number of commits (over a seven hundred each); and they were developed using an object oriented programming language (Java).

This paper is structured as follows: Section 2 presents the concepts related to Code Smells and the classification of the developers. Section 3 describes a proposed method to sort the developers in groups and assess the contribution on the variation of Code Smells in the source code of the software. Section 4 demonstrates a case study for the application of the method of classification of developers, evaluating the influence of each developer group in variation of *Code Smells*. Section 5 describes related work and finally, Section 6 presents the conclusion of this article.

II. STUDY PRELIMINARIES

This Section presents the definitions of code smells and of developer characteristics used in our study.

A. Code Smells

Webster [8] defined *antipatterns* in object-oriented development. An antipattern is similar to a pattern except it is an obvious but wrong solution to a problem. Nevertheless, these antipatterns will be tried again by someone simply because they appear to be the right solution [9]. *Code smells* refer to structural characteristics of a source code that indicate this code has problems, affecting directly on the maintainability of the software and resulting in a greater effort to carry out developments in this source code [10].

 ISBN: 978-1-61208-438-1

B. Developer Characteristics

Software development is a human activity [11]. Understanding the human factors of the developers allows software managers to organize them in groups, so that they can compose more efficient teams [12]. Whereas distinguishing and verifying the impacts of each developer individually is a very difficult task, developers can be categorized according to their involvement in a software project. The involvement of a developer can be measured in terms of level of participation and degree of authorship on the source code [13].

The level of participation is related to the developer's involvement in the project and can be used, for example, to determine the degree of decision-making the developer has in the project team, allowing discover developers who exercise leadership in project [13][14]. The degree of authorship indicates the usual tasks the developer performs when acting on the software source code. It involves line code change, insertion or removal and file (e.g., class in an object-oriented system) insertion and removal.

III. Study Settings

The goal of our study is to investigate the influence of developers on the introduction of code smells in a software code. To do so, we analyzed the sequence of commits done in the repository of five different software projects. Merge (branches) were considered in the selection of the project commits.

First, we categorized the developers in different groups according to their characteristics in the project (participation and authorship). Then, for each commit, we searched for code smells in the source code. The quality focus was the analysis on the variation of the number of code smells along the time.

To categorize the developers, we used the k-means clustering algorithm [15]. The information used in the k-means algorithm was taken from the software repository and they were related to the participation level and degree of authorship of the developers in each selected project.

To find code smells in the source code, we used PMD [16], a static rule-set based Java source code analyser that seeks to evaluate aspects related to good programming practices.

A. System Characteristics

The first decision we made in our study was the selection of the target systems. We chose five medium-size systems. The first one, called Behave, is an automation tool for functional testing. It was first versioned in 2013 and we found 724 commits in its project. We selected 373 commits of Behave in our study. The second was JUnit, a unit-testing framework for the Java programming language. It was first versioned in 2000 and we found 1885 commits in its project. We selected 1203 commits of Junit in our study. The third one was Mockito, an open source-testing framework for Java, which allows the creation of test double objects (mock objects) in automated unit tests for the purpose of Test-driven Development or Behavior Driven Development. It was first versioned in 2007 and we found 1993 commits in its project. We selected 1561 commits of Mockito in our study. The fourth one, called RxJava, is a library for composing asynchronous and event-based programs using observable sequences for the Java VM. It was first versioned in 2012 and we found 2939 commits in its project. We selected 906 commits of RxJava in our study. The last system was VRaptor, a Java MVC Framework focused in delivering high productivity to web developers. It was first versioned in 2009 and we found 3385 commits in its project. We selected 2243 commits of VRaptor in our study. The projects selected for this study were taken from the Git repository on June 2014.

These systems were chosen because they met a number of relevant criteria for our study. First, these systems encompass a rich set of code smells (e.g., Dead Code, Long Method, Unhandled Exception). Second, they are non-trivial systems and their sizes are manageable for an analysis of code smells. Third, each one of them were implemented by more than 50 programmers with different levels of participation (the selected systems were all open source projects). Last, they have a significant lifetime, comprising of several commits. The availability of multiple commits allowed us to observe the introduction of code smell throughout their long-term development and evolution.

It should be noted that for this study, commits were discarded that altered documentation of source code, HTML pages and templates (css, imagens, javascript) changes because they have no relation with change of code smells.

B. Study Phases

Our study was based on the analysis of the developers' information and the systems' code smells. The main phases of our study are described next.

Recovering the Developers' Information. In this phase, we focused in gathering information about the level of participation and degree of authorship of a developers. The reason was that we needed to group the developers so that we could rely on general coding behaviour instead of trying to focus in each developer separately. We selected information from the data available in the Git repository. As a result, we concentrated on the analysis of information for each developer commit. For level of participation, we collect date and time of commit initial, date and time of last commit and interval (days) between commits. For degree of authorship, we collect amount of modified files (classes) in the commit (insertion, modification and deletion) and the amount of lines of code modified during the commit (insertion and deletion).

Classifying the Developers. The recovered information was used in the k-means clustering algorithm to identify groups of developers with similar characteristics, according to their participation and degree of authorship in the project. In this study, we used the k-means algorithm varying the value of k from four up to nine in order to verify the distribution of developers in the clusters.

The overall results provided six sets of developer clusters. Analysing these sets, we decided to use the results from k=5 (five clusters) because we wanted to avoid the presence of very scarce clusters. Then we used the apriori association algorithm to find correlations between different attributes in each cluster. The results identified the general association rules for the population of each cluster, as shown in Table I.

(a) *Group 01:* less frequent participation and line code deletion as general authorship behaviour;

(b) *Group 02:* less frequent participation and line code insertion and deletion as general authorship behaviour;

 ISBN: 978-1-61208-438-1

(c) *Group 03:* less frequent participation and file insertion, modification and deletion as general authorship behaviour;

(d) *Group 04:* more frequent participation and no particular general authorship behaviour (i.e., it performs all behaviours almost evenly);

(e) *Group 05:* more frequent participation and file insertion as general authorship behaviour.

TABLE I. POPULATION OF EACH CLUSTER

Gr.	Behave	JUnit	Mokito	RxJava	VRaptor
01	2	15	5	2	24
02	7	27	25	31	10
03	10	33	58	13	31
04	2	14	17	17	15
05	15	6	5	10	13

Selection of Code Smells. In this phase, we focused on selecting code smells that were previouly described in the literature. Moreover, we have not considered creating specific PMD rulesets to identify code smells. The reason was that we needed to rely on code smells that could be precisely identified in a systematic fashion, without any specialist assistance. As a result, we concentrated on the analysis of five existing code smells [17], which covered various anomalies related to object oriented programming. Those were: Dead Code (DC); Large Class (LC); Long Method (LM); Long Parameter List (LPL); and, Unhandled Exception (UE).

Identifying Occurrences of Code Smells. Code smells were identified using the PMD tool. Thus, code smells were detected using five ready-to-run PMD rulesets. We decided not to define specific rules for this study because we understand that code smells should be identified as simply as possible.

Analysis of Code Smell Introduction. The goal of the fifth phase was to analyse the behaviour of code smell introduction for the selected projects. The analysis aimed at triggering some insights for helping maintainers to understand the relationships between code smell introduction and the developers in the project team. To support the data analysis, the assessment phase was decomposed in three main stages. The first stage aimed at examining the occurrence frequency of each code smell in the analyzed commits. The second stage was concerned with observing the participation of the developers in the analyzed commits. The last stage focused on assessing the relationship of developers on a code smell manifestation. In this last stage, we calculated the average percentage of introduction and of removal of the selected code smell by each group of developers. The idea is to verify the general influence of each group in the project.

IV. STUDY FINDINGS

The first subsection below shows the total number of each investigated code smell in the target systems. The following five subsections report the findings associated with the characterization of code smells and the involvement of the developers. Finally, the last subsection presents some discussion about the results and the impact of developers in code smell introduction.

A. Occurrence of Code Smells

There was a significant difference on how often each investigated code smell occurred in the target systems. The results are summarized in Table II. The "I" column indicates the total number of times each code smell was inserted in each target system and the "R" column indicates the total number of times each code smell was removed. The "Tot" line presents the total number of smell instances detected (inserted and removed respectively). For "I" equal to 0 means that there was no inclusion of this code smell. For "R" equal to 0, means that no removal of said code smell. It is important to mention that not all code smells inserted in the analyzed commits were removed.

TABLE II. CODE SMELL OCCURRENCES

	Behave		JUnit		Mokito		RxJava		VRaptor	
CS	I	R	I	R	I	R	I	R	I	R
DC	91	81	208	262	230	335	168	224	311	517
LC	39	92	204	242	181	340	215	224	220	403
LM	46	61	98	161	112	353	149	225	150	409
LPL	0	0	3	9	0	0	63	112	0	1
UE	69	95	240	269	337	288	205	236	377	419
Tot	245	329	753	943	860	1316	800	1021	1058	1749

B. Dead Code

The *Dead Code* code smell refers to code that is not been used. These code smells were identified using the Empty Code, Unnecessary and Unused Code rulesets in PMD. These rulesets are composed of the following rules:

(a) *Empty Code:* this ruleset aims to check if there are empty statements of any kind (empty method, empty block statement, empty try or catch block, etc.);

(b) *Unnecessary:* this ruleset aims to determine whether there are unnecessary code (unnecessary returns, final modifiers, null checks, etc.);

(c) *Unused Code:* this ruleset aims to find unused or ineffective code (unused fields, variables, parameters, etc.).

The results are summarized in Table III, which shows the percentage of insertion and removal of the Dead Code smell for each target system by developer group.

TABLE III. RESULTS FOR DEAD CODE

	Behave		JUnit		Mokito		RxJava		VRaptor	
Gr.	I%	R%	I%	R%	I%	R%	I%	R%	I%	R%
01	0	0	0	0	74	78	40	39	0	0
02	76	67	25	31	10	12	9	9	15	17
03	24	33	58	53	3	3	29	32	34	26
04	0	0	17	17	0	0	0	0	51	56
05	0	0	0	0	13	7	22	20	0	0

C. Large Class

The *Large Class* code smell refers to classes that are trying to do too much, often showing up as too many instance variables. These code smells were identified using a subset of the Code Size ruleset in PMD. The rules used to identify this code smell were:

 ISBN: 978-1-61208-438-1

(a) *Excessive Class File Length:* usually indicates that the class may be burdened with excessive responsibilities that could be provided by external classes or functions;

(b) *Excessive Public Count:* seeks for large numbers of public methods and attributes.

(c) *NCSS Type Count:* uses the NCSS (Non-Commenting Source Statements) algorithm to determine the number of lines of code for a given type;

(d) *Too Many Fields:* determines if a class has too many fields in its code;

(e) *Too Many Methods:* determines if a class has too many methods in its code.

The results are summarized in Table IV, which shows the percentage of insertion and removal of the Large Class smell for each target system by developer group.

TABLE IV. RESULTS FOR LARGE CLASS

	Behave		JUnit		Mokito		RxJava		VRaptor	
Gr.	I%	R%	I%	R%	I%	R%	I%	R%	I%	R%
01	0	0	0	0	70	79	35	37	0	0
02	69	72	28	29	15	10	10	8	19	15
03	31	28	58	53	2	4	34	30	39	34
04	0	0	14	18	0	0	0	0	42	51
05	0	0	0	0	12	7	20	25	0	0

D. Long Method

The *Long Method* code smell refers to methods that are trying to do too much, often presenting too much code. These code smells were identified using a subset of the Code Size ruleset in PMD. The rules used to identify this code smell were:

(a) *Excessive Method Length:* seeks for methods that are excessively long;

(b) *NCSS Method Count:* uses the NCSS algorithm to determine the number of lines of code for a given method;

(c) *NCSS Constructor Count:* uses the NCSS algorithm to determine the number of lines of code for a given constructor;

(d) *NPath Complexity:* determines the NPath complexity of a method (the number of acyclic execution paths through that method).

The results are summarized in Table V, which shows the percentage of insertion and removal of the Long Method smell for each target system by developer group.

TABLE V. RESULTS FOR LONG METHOD

	Behave		JUnit		Mokito		RxJava		VRaptor	
Gr.	I%	R%	I%	R%	I%	R%	I%	R%	I%	R%
01	0	0	0	0	71	77	42	36	0	0
02	72	64	24	29	13	12	8	10	16	16
03	28	36	58	43	4	3	28	30	39	25
04	0	0	17	28	0	0	0	0	45	60
05	0	0	0	0	13	8	23	24	0	0

E. Long Parameter List

The *Long Parameter List* code smell refers to methods that present a long parameter list usually involving global data. These code smells were identified using a single rule of the Code Size ruleset in PMD:

(a) *Excessive Parameter List:* seeks for methods with numerous parameters.

The results are summarized in Table VI, which shows the percentage of insertion and removal of the Long Parameter List smell for each target system by developer group.

TABLE VI. RESULTS FOR LONG PARAMETER LIST

	Behave		JUnit		Mokito		RxJava		VRaptor	
Gr.	I%	R%	I%	R%	I%	R%	I%	R%	I%	R%
01	0	0	0	0	0	0	40	35	0	0
02	0	0	33	67	0	0	14	19	0	0
03	0	0	67	22	0	0	22	26	0	0
04	0	0	0	11	0	0	0	0	0	0
05	0	0	0	0	0	0	24	21	0	0

F. Unhandled Exceptions

The *Unhandled Exceptions* code smell refers to pieces of code containing malformed throw/try/catch statements. These code smells were identified using a single ruleset in PMD:

(a) *Strict Exceptions:* provides some strict guidelines about throwing and catching exceptions.

The results are summarized in Table VII, which shows the percentage of insertion and removal of the Unhandled Exceptions smell for each target system by developer group.

TABLE VII. RESULTS FOR UNHANDLED EXCEPTIONS

	Behave		JUnit		Mokito		RxJava		VRaptor	
Gr.	I%	R%	I%	R%	I%	R%	I%	R%	I%	R%
01	0	0	0	0	71	82	38	39	0	0
02	71	73	27	33	15	8	13	7	16	14
03	29	27	58	49	3	2	25	33	31	33
04	0	0	15	18	0	0	0	0	53	53
05	0	0	0	0	11	8	24	21	0	0

G. Discussion

Tables 2 to 6 presented the percentage of participation of each group of developers in the insertion and removal of code smells for the five studied systems, represented by the $\%I$ column and the $\%R$, respectively. In the analyzed set of commits of the Behave system, in general, groups 2 and 3 were responsible for inserting and removing such code smells. Group 2 inserted more smells but also removed in an even proportion. For JUnit and VRaptor, groups 2, 3 and 4 were responsible for inserting and removing code smells. Four groups inserted and removed code smells in the Mokito system, but the results point out to group 1 as been the one group with more impact on the insertion and removal of code smells. Finally, for the RxJava, the results indicate that groups 1, 3 and 5 were more responsible for inserting and removing code smells.

 ISBN: 978-1-61208-438-1

In the selected set of commits analyzed in this study, all code smells were decreased (had more removals than insertions). This is an indication that the occurrence of code smells depends on the software evolution. It seems that the code smells in the study tend to appear in preliminary releases with more frequency. We did not use the initial commits in our study to avoid the "cold start" problem as we believed these data would not have a proper indication of code smell removal.

Code Smells with Highest Frequencies. The code smells associated with the problem of dead code and unhandled exceptions fell in the group of highest insertion frequency for the analyzed target systems. A closer look made us to suspect that this probably occurred because groups 1 and 2 were more involved in these code smells. Such groups do not present a high level of participation and have a common authorship behavior, which is line code removal. We understand that, in some cases, lines may have been removed without the appropriate care, resulting in dead code and unhandled exceptions. The code smells associated with the problem of long method fell in the group of highest removal frequency for the analyzed target systems. This finding suggests that the development team for the target systems may have done proper refactoring as to decrease the size of the methods.

No Influence on Code Smells. The classification process found members for all groups in the development teams of every target system. However, there were groups that were not involved with code smells in some systems. For instance, group 4 did not insert nor remove code smells in the Mokito system. Groups 1, 4 and 5 did not insert nor remove code smells in the Behave system. We suspect that this occurred because there were few members in these groups for such systems. We used the whole dataset to classify the developers and when we took a deeper look in a system by system basis, some groups were scarce.

Developers vs. Code Smells. In general, groups 1 to 3 (groups whose members have fewer participation in the code development) tended to have a higher engagement in the introduction and removal of code smells. Initially, we thought that the developers in the groups with higher participation frequency would have more impact in code smell removal. This was not observed. We believe that, in the context of our study, this may have happened due to the fact that groups 4 and 5 were more more responsible for in adding functionality to the target systems whereas the other groups were more involved in fault correction.

Recommendations. Considering the results, it is necessary to evaluate the quality of the source code, taking into account the inclusion and removal of problematic code snippets. Thus, the developers assessment process (Group) must be reevaluated constantly, based on data related to the project's commit history. In addition, it is recommended that there is a mixture of different groups, considering the features that contribute to remove code smells.

H. Limitations

Some limitations or imperfections of our study can be identified and are discussed in the following.

Construct Validity. Threats to construct validity are mainly related to possible errors introduced during specific data processing from the repository. The repository did not provide an unique identification data for a developer, thus, it was not possible to determine whether a developer performed commits with different identifications. In this sense, each developer (responsible) identified in the repository was treated as a different developer. However, the study was not intended to focus on the contribution of a specific developer.

Conclusion Validity. We have three issues that threaten the conclusion validity of our study: the number of evaluated systems; the evaluated code smells (and their relation to the PMD rules), and; discarding the data from the commits that did not increase nor decrease the number of code smells. Five open source projects from Git were analyzed. A higher number of systems is always desired. However, the analysis of a bigger sample in this kind of study could be non-practical. The number of systems with all the required information available to perform this kind of study is bare. We understand that our sample can be seen as appropriate for a first exploratory investigation [18]. Related to the second issue, our analysis used the PMD tool. Regarding the set of code smells used in the study, code smells reported in the literature were considered in our study. Finally, we discarded data from commits that maintained the amount of code smells. Although the study focused on associating developer profiles to improving or lessen the quality of the code, we understand that this limitation does not allow us to make a conclusion for a specific code smell.

V. Related Work

There are several approaches available in the literature for detecting Code Smells. Mantyla investigated as developers identify and treat Code Smells in the source code to compare with automated detection methods [19]. There are also several approaches available in the literature for investigation of the effects of Code Smells in aspects related to software maintainability [20], such as defects [21], effort [22] and requests for changes [23]. In addition, few studies have focused on the detection of Code Smell through mining activities in software repositoryc [24].

Regarding the classification of developers in groups, there are several existing approaches in the literature. In this context, one of the proposals is based on data extracted from the repository in relation to the time of performing the commit. Thus, the model proposes to assess in which the range of hours developers insert more bugs in your commits [25]. Another approach is to sort the developers on the basis of the records related to quantity, time, and type of actions and activities that these developers come true, working on the project, and the data extracted from the version control system and other tools, such as mailing list and bug tracker tools [26][27].

VI. Concluding Remarks

This work presented a study to assess the influence of developers on the introduction of code smells in a software system. We classified the developers into five categories and verified their contributions (increasing or decreasing) in the number of code smells in a set of consecutive software versions. This exploratory study revealed, within the limits of the threats to its validity, the conjecture that the team member behaviour (participation frequency, authorship and development activity - feature development or fault correction) impacts in the insertion and removal of code smells.

 ISBN: 978-1-61208-438-1

Finally, it is important to highlight that we have analyzed commits of five systems. Then, the relationships of code smells and developers should be tested in broader contexts in the future. It would also be desirable to use the development activity of the developers in the classification and association of developers.

ACKNOWLEDGMENT

The authors thank everyone who provided knowledge and skills that really helped the search. The result is a compilation of ideas and concepts throughout the development of this work.

REFERENCES

[1] IEEE, IEEE Standard for Software Maintenance, IEEE Std 1219-1998. IEEE Press, 1999, vol. 2.

[2] "Software engineering - product quality, ISO/IEC 9126-1," International Organization for Standardization, Tech. Rep., 2001.

[3] F. A. Fontana and M. Zanoni, "On investigating code smells correlations," in ICST Workshops'11, 2011, pp. 474–475.

[4] A. F. Yamashita and L. Moonen, "Do developers care about code smells? an exploratory survey." in WCRE, R. Lammel, R. Oliveto, and R. Robbes, Eds. IEEE, pp. 242–251. [Online]. Available: http://dblp.uni-trier.de/db/conf/wcre/wcre2013.html (access date: September 2015)

[5] I. M. Bertran, "Detecting architecturally-relevant code smells in evolving software systems," in Proceedings of the 33rd International Conference on Software Engineering, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 1090–1093. [Online]. Available: http://doi.acm.org/10.1145/1985793.1986003 (access date: September 2015)

[6] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, and G. Succi, "Balancing agility and formalism in software engineering," B. Meyer, J. R. Nawrocki, and B. Walter, Eds., 2008, ch. A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team, pp. 252–266.

[7] GitHub, "Git repository," https://github.com, 2014.

[8] B. F. Webster, Pitfalls of object-oriented development. M And T, 1995.

[9] J. Long, "Software reuse antipatterns," SIGSOFT Softw. Eng. Notes, vol. 26, no. 4, Jul. 2001, pp. 68–76. [Online]. Available: http://doi.acm.org/10.1145/505482.505492 (access date: September 2015)

[10] J. Schumacher, N. Zazworka, F. Shull, C. Seaman, and M. Shaw, "Building empirical support for automated code smell detection," in Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ser. ESEM '10. New York, NY, USA: ACM, 2010, pp. 8:1–8:10. [Online]. Available: http://doi.acm.org/10.1145/1852786.1852797 (access date: September 2015)

[11] R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th ed. McGraw-Hill Higher Education, 2001.

[12] E. Di Bella, A. Sillitti, and G. Succi, "A multivariate classification of open source developers," Inf. Sci., vol. 221, Feb. 2013, pp. 72–83. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2012.09.031 (access date: September 2015)

[13] S. Matsumoto, Y. Kamei, A. Monden, K.-i. Matsumoto, and M. Nakamura, "An analysis of developer metrics for fault prediction," in Proceedings of the 6th International Conference on Predictive Models in Software Engineering, ser. PROMISE '10. New York, NY, USA: ACM, 2010, pp. 18:1–18:9. [Online]. Available: http://doi.acm.org/10.1145/1868328.1868356 (access date: September 2015)

[14] M. Zhou and A. Mockus, "Developer fluency: achieving true mastery in software projects," in Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7-11, 2010, 2010, pp. 137–146. [Online]. Available: http://doi.acm.org/10.1145/1882291.1882313 (access date: September 2015)

[15] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297.

[16] InfoEther, "Pmd is a source code analyzer," http://pmd.sourceforge.net/, 2014.

[17] M. Fowler, Refactoring: Improving the Design of Existing Code. Boston, MA, USA: Addison-Wesley, 1999.

[18] B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu, "Evaluating guidelines for empirical software engineering studies," in Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, ser. ISESE '06. New York, NY, USA: ACM, 2006, pp. 38–47. [Online]. Available: http://doi.acm.org/10.1145/1159733.1159742 (access date: September 2015)

[19] M. Mantyla and C. Lassenius, "What types of defects are really discovered in code reviews?" IEEE Trans. Software Eng., vol. 35, no. 3, 2009, pp. 430–448. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TSE.2008.71 (access date: September 2015)

[20] D. I. Sjoberg, A. Yamashita, B. C. Anda, A. Mockus, and T. Dyba, "Quantifying the effect of code smells on maintenance effort," IEEE Transactions on Software Engineering, vol. 39, no. 8, 2013, pp. 1144–1156.

[21] F. Rahman, C. Bird, and P. T. Devanbu, "Clones: What is that smell?" in MSR, J. Whitehead and T. Zimmermann, Eds. IEEE, 2010, pp. 72–81. [Online]. Available: http://dblp.uni-trier.de/db/conf/msr/msr2010.html (access date: September 2015)

[22] M. Abbes, F. Khomh, Y.-G. Gueheneuc, and G. Antoniol, "An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension." in CSMR, T. Mens, Y. Kanellopoulos, and A. Winter, Eds. IEEE Computer Society, 2011, pp. 181–190. [Online]. Available: http://dblp.uni-trier.de/db/conf/csmr/csmr2011.html (access date: September 2015)

[23] S. M. Olbrich, D. Cruzes, and D. I. K. Sjberg, "Are all code smells harmful? a study of god classes and brain classes in the evolution of three open source systems." in ICSM. IEEE Computer Society, 2010, pp. 1–10. [Online]. Available: http://dblp.uni-trier.de/db/conf/icsm/icsm2010.html (access date: September 2015)

[24] R. Peters and A. Zaidman, "Evaluating the lifespan of code smells using software repository mining," in Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on. IEEE, 2012, pp. 411–416.

[25] J. Eyolfson, L. Tan, and P. Lam, "Do time of day and developer experience affect commit bugginess?" in Proceedings of the 8th Working Conference on Mining Software Repositories, ser. MSR '11. New York, NY, USA: ACM, 2011, pp. 153–162. [Online]. Available: http://doi.acm.org/10.1145/1985441.1985464 (access date: September 2015)

[26] W. Poncin, A. Serebrenik, and M. van den Brand, "Process mining software repositories," in Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on, March 2011, pp. 5–14.

[27] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye, "Evolution patterns of open-source software systems and communities," in Proceedings of the International Workshop on Principles of Software Evolution, ser. IWPSE '02. New York, NY, USA: ACM, 2002, pp. 76–85. [Online]. Available: http://doi.acm.org/10.1145/512035.512055 (access date: September 2015)

 ISBN: 978-1-61208-438-1

The Object Oriented Petri Net Component Model

Radek Kočí and Vladimír Janoušek

Brno University of Technology, Faculty of Information Technology,
IT4Innovations Centre of Excellence
Czech Republic
email: {koci,janousek}@fit.vutbr.cz

***Abstract*—The formalism of Object Oriented Petri Nets (OOPN) is a part of the work dealing with the method of system development in simulation. The work is based on the idea that system models are always executed even if they contain only one simple element or any changes are performed. Moreover, this idea does not distinguish between system models, prototypes, or target system; everything should be presented by the same means. Nevertheless, it should be possible to use different formalisms to describe models. It follows that a common platform is needed. The platform has to be simple and has to allow to change models on the fly. The formalism of Discrete Event System Specification (DEVS) has been used to specify the platform, because it enables to compose system using components, whereas each such a component can be modeled by different formalism. Proposed approach preserves the advantages of using OOPN for behavior modeling of components and makes it possible to hierarchize models using DEVS-based platform. The paper defines a platform based on DEVS and OOPN formalisms and deals with a question of safe changes of components on the fly.**

Keywords–Object Oriented Petri Nets; DEVS; component platform; interface consistency.

I. INTRODUCTION

This paper is part of the *System in Simulation Development* (SiS) work [1] based on the formalism of Object oriented Petri nets (OOPN) [2]. The basic SiS principle consists in continuous incremental development of models in the *live system* with the goal to come to the target system without a need of implementation—there is no difference between models, prototypes, or target system. The SiS concept requires three basic conditions. First, models have to be able to combine different formalisms or languages, e.g., Petri nets and Smalltalk language. For instance, the control part of the developed system can be modeled by OOPN, which has to be able to communicate to sensors—the communication channel can be implemented in Smalltalk language. Second, models can be execute in different simulation modes that are suitable for design, testing, in-the-loop simulation, and system deployment [3]. Third, there has to be a possibility to exchange any elements of the models on the fly; the model elements should be exchanged with no changes in the depending model elements [4].

To achieve presented requirements, a *common platform* is needed. The platform has to be simple and has to fulfill the SiS requirements, mainly changing models on the fly. The formalism of Discrete Event System Specification (DEVS) has been used to specify the common platform. It enables to compose system using DEVS-based components, whereas each such a component is modeled by means of OOPN. It preserves the advantages of using OOPN for behavior modeling and makes it possible to hierarchize models.

So far, there have been works dealing with a usage of OOPN and DEVS formalisms, but the compact definition of common platform has not been introduced and a question about safe replacement has not been solved. The paper defines the OOPN component model based on the DEVS common platform to which the formalism of OOPN is incorporated. The question about component interfaces and their consistency during the component changes will also be discussed.

The paper is organized as follows. We describe concepts of the common platform in Section III. Then, we define the OOPN component model based on the common platform in Section IV. The Section V describes a problem of the component interface consistency and introduces interface constraints. Section VI deals with a realization of constraints based on the formalism of OOPN. The summary and future work is described in Section VII.

II. RELATED WORK

The modeling of software system in *live* environment is not new idea. Model-Driven Software Development [5][6] uses executable models, e.g., Executable UML [7], which allows to test systems using models. Models are then transformed into code, but the resulted code has to often be finalized manually and the problem with imprecision between models and transformed code remains unchanged. Further similar work based on ideas of model-driven development deals with gaps between different development stages and focuses on the usage of conceptual models during the simulation model development process—these techniques are called *model continuity* [8]. While it works with simulation models during design stages, the approach proposed in this paper focuses on *live models* that are used in target environments, i.e., when the system is deployed.

The research activities in the area of system changes on the fly are usually focused on direct or indirect approaches. The direct approach uses formalisms containing intrinsic features allowing to change the system. Formalisms are usually based on kinds of Petri nets. Reconfigurable Petri Nets [9] introduces a special place describing the reconfiguration behavior. Net Rewriting System [10] extends the basic model of Petri nets and offers a mechanism of dynamic changes description. This work has been improved [11] by a possibility to implement net blocks according to their interfaces. Intelligent Token Petri Nets [12] introduces tokens representing jobs by that the

 ISBN: 978-1-61208-438-1

dynamic changes can be easily modeled. Their disadvantage is that they usually do not define the modularity.

The indirect approach handles system changes using extra mechanisms. Model-based control design method, presented by Ohashi and Shin [13], uses state transition diagrams and general graph representations. Discrete-event controller based on finite automata has been presented by Liu and Darabi [14]. The presented methods use external mechanisms, nevertheless, most of them do not deal with validity of changes.

The approach presented in this paper combines direct and indirect methods. To define platform allowing to change component on the fly, the intrinsic features of the formalism of DEVS is used in combination with application framework allowing to work with simulation in live environment.

III. COMMON PLATFORM

As we mentioned above, we need to have a *common platform* allowing to interconnect different formalisms, as well as to change model element on the fly. We have decided [4] to use DEVS [15] approach to specify the platform. This section describes a formal base of the common platform and introduces a simple example to demonstrate its features and usage.

A. Discrete Event System Specification Platform

The formalism of DEVS can represent any system whose input/output behavior can be described as a sequence of events. The model consists of *atomic models* M. Their behavior is described by functions that work with input event values X and produce output event values Y. These functions are not important from the paper point of view, so that we will abstract them. Atomic models can be coupled together to form a *coupled model* CM. The later model can itself be employed as a component of a larger model. The atomic model, as well as the coupled model, corresponds to the term *component*. This way the DEVS formalism brings a hierarchical component architecture. The platform based on DEVS will be called *common component platform* and will be denoted $\mathcal{M}$. The set of components of the platform $\mathcal{M}$ will be denoted $\mathcal{D}$.

B. Component Interface

Sets X and Y of the component are usually specified as structured sets. It allows to define input and output ports for input and output events specification, as well as for coupling specification. Let us have the structured set $X = \{(v_1, v_2, \ldots, v_n) | v_1 \in X_1, \ldots, v_n \in X_n)\}$, where v_i represents a value of the ith variable from the domain set X_i. We will denote members $v_1, v_2, \ldots, v_n$ as *input ports* and will write $X = (V_X, X_1 \times X_2 \times \cdots \times X_n)$, where V_X is an ordered set of n *input ports*. The set of *output ports* V_Y is defined similarly on the structured set Y. The *component interface* is then built up from *input ports* V_X and *output ports* V_Y.

The component platform consists of components that are coupled through their ports. We define a relationship *coupling* $\xrightarrow{\mathcal{D}} \subseteq \bigcup_{i\in\mathcal{D}} V_Y^i \times \bigcup_{i\in\mathcal{D}} V_X^i$ meaning that there are channels for data transmission between ports of components. We will denote input port, resp. output port, by the notation $component\_name \oplus port\_name$, resp. $component\_name \ominus port\_name$. Then, the notation $c_1 \ominus p_1 \xrightarrow{\mathcal{D}} c_2 \oplus p_2$ means that there is the coupling between the output port p_1 of the component c_1 and the input port p_2 of the component c_2. The relationship $\xrightarrow{\mathcal{D}}$ can also be written in opposite direction $\xleftarrow{\mathcal{D}}$.

Then, the common component platform is defined $\mathcal{M} = (\mathcal{D}, \xrightarrow{\mathcal{D}}, V_X^{\mathcal{M}}, V_Y^{\mathcal{M}})$, where $V_X^{\mathcal{M}} = \bigcup_{i\in\mathcal{D}} V_X^i$ and $V_Y^{\mathcal{M}} = \bigcup_{i\in\mathcal{D}} V_Y^i$ represent ports that are accessible from the platform neighborhood.

C. Component Changes on the Fly

The component in the common platform is a model description, as well as its executable form. There is no difference between static and dynamic (live) representation of models. In comparison with classic object oriented approach, we need not care about classes, new instances, and reference changes (i.e., how to detach old objects and to attach new objects) at the moment of component changes. We simply create the new component and change the connections (couplings).

D. Example Specification

The concepts presented in the paper will be demonstrated on the small example consisting of sensor nets, a module collecting data from sensors, and a module making decision based on the data (the form of decision is not important). Other parts will be abstracted.

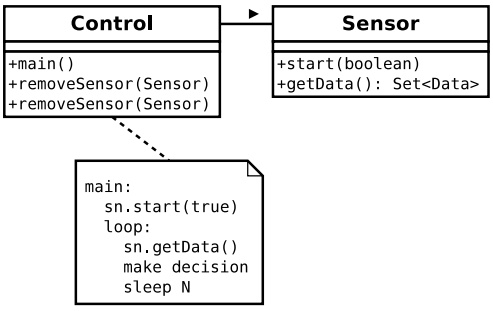


Figure 1. An example—class diagram.

In the classic object approach, we can define two analytical classes *Control* (decision maker) and *Sensor* (data collector). The class *Sensor* defines operations allowing to start or stop the data collection (*start*) and to get acquired data from sensors (*getData*). The class *Control* defines operations to attach and to detach a sensor (*addSensor* and *removeSensor*) and to control the process (*main*). The class diagram is shown in Figure 1. The basic algorithm of the method *main* is illustrated in the note window—it starts data collection and then performs following operations in the loop: gets data, makes decision, and waits for a while.

Now, we take the example specification to the common component platform. Figure 2 shows an example of the platform $\mathcal{M}_1$ containing two components $Control$ and $Sensors$, where $Sensors$ represents a communication channel to the sensor nets and $Control$ receives acquired data and makes decisions about the system. Due to simplification of notation, we will write cn and sn instead of full names.

Then, the platform consists of $\mathcal{D}_1 = \{cn, sn\}$, where the components interfaces consists of $V_X^{cn} = \{data, run, stop\}$, $V_Y^{cn} = \{start, request\}$, $V_X^{sn} = \{start, request\}$, and $V_Y^{sn} = \{data\}$.

Now, we can compare DEVS-based common platform with object approach. First, the class $Sensor$ and the component sn. The method $start$ is represented by the input port $sn \oplus start$ receiving a command to start or stop data collecting. The method $getData$ is represented by a pair of input port $sn \oplus request$ and output port $sn \ominus data$. If any component asks for data, it puts a command to $sn \oplus request$ and the component reacts by putting data to $sn \ominus data$.

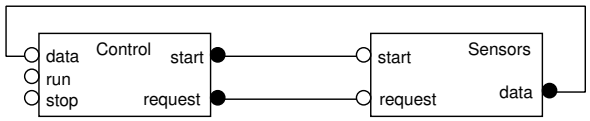


Figure 2. Common platform $\mathcal{M}_1$.

Second, the class $Control$ and the component cn. There is no port equivalent to the method $main$ because of intrinsic definition of the component behavior. Nevertheless, ports $cn \oplus run$ and $cn \oplus stop$ serve to start and to stop main loop of the component cn. These ports are not connected inside the platform; they will be used from outside to control platform run. The communication to cn surroundings is represented by ports $cn \ominus start$ (starting a data collection), $cn \ominus request$ (a request for acquired data) and $cn \oplus data$ (an answer for data requesting). The component cn sends commands to the component sn by carrying data through $cn \ominus start \xrightarrow{\mathcal{D}} sn \oplus start$ and $cn \ominus request \xrightarrow{\mathcal{D}} sn \oplus request$. The component sn reacts by sending data through the coupling $cn \oplus data \xleftarrow{\mathcal{D}} sn \ominus data$.

IV. OOPN COMPONENT MODEL

The common platform based on DEVS formalism offers component approach allowing to wrap another kind of formalisms, so that each such a formalism is evaluated by own means. The Object oriented Petri Net component model (OOPN component model) consists of DEVS components that are described by the OOPN formalism. This section introduces the OOPN formalism and its relationship to the *common platform* $\mathcal{M}$.

A. Object Oriented Petri Nets

First of all, let us agree upon the following definitions in the OOPN component system. The *Object oriented Petri net* is a tuple (Σ, c_0), where Σ is a system of classes and c_0 is an initial class. Σ contains sets of OOPN elements, which constitute classes. For the paper purpose, we will denote only selected elements that are used. The system of classes Σ is defined as follows $\Sigma = (C_{PN}, MSG, N_O, N_M, SP, NP, P, T)$, where C_{PN} is a set of OOPN classes, MSG is a set of message selectors, N_O is a set of object nets, N_M is a set of method nets, SP is a set of synchronous ports, NP is a set of negative predicates, P is a set of places, and T is a set of transitions. The message selectors MSG corresponds to method nets, synchronous ports, and negative predicates. Object nets describe possible autonomous activities of objects, while method nets describe reactions of objects to messages. A *class* C is defined as $C = (MSG^C, on^C, N_M^C, SP^C, NP^C)$, where $MSG^C \subseteq MSG$, $on^C \in N_O$, $N_M^C \subseteq N_M$, $SP^C \subseteq SP$, and $NP^C \subseteq NP$. Every net consists of places (a subset of P) and transitions (a subset of T).

The OOPN dynamics comprises the system of objects Γ. Elements from C describe a structure of simulation model and have to be instantiated to simulate the model. If the class $C \in C_{PN}$ is instantiated (the object o is created), the instance of object net on^C is created immediately. If the message $m \in MSG$ is sent to the object o, an instance of the method net is created. Then, we can define $\Gamma = (OBJ, INV)$, where OBJ is a set of objects including their object net instances and INV is a set of invoked method nets.

B. OOPN in Common Platform

In the common platform, there is no difference between a static representation of the model and its *live* (running, executed) form. To include the OOPN formalism, we introduce the *live model* of OOPN as the tuple $\Pi = (\Sigma, \Gamma, c_0, obj_0)$, where $c_0 \in C_{PN}$ is an initial class and $obj_0 \in OBJ$ is an initial object of the class c_0.

In the common platform, the OOPN model is split up into submodels, whereas each submodel has its own initial class c_0 and initial object obj_0. Let $M_{PN} = (M, \Pi, P_{c0}^{inp}, P_{c0}^{out})$ be a DEVS component M, which wraps an OOPN submodel Π. The initial class c_0 is instantiated immediately the component M_{PN} is created. The component interface (V_X, V_Y) is represented by subsets of places $P_{c0}^{inp}, P_{c0}^{out} \subseteq P$, where P is a set of object net places of the initial class c_0 and $P_{c0}^{inp} \cap P_{c0}^{out} = \emptyset$. There are bijections $map_{inp} : P_{c0}^{inp} \to V_X$ and $map_{out} : P_{c0}^{out} \to V_X$ mapping ports and places and the mapped places then serve as input or output ports of the component.

C. OOPN Example

Let us continue with the example from Figure 2. Figure 4 shows an OOPN model of the component *Sensors* (sn) and Figure 3 shows an OOPN model of the component *Control* (cn). Both models have the same basis—the loop driven by an external stimulus (a token placed in the place s).

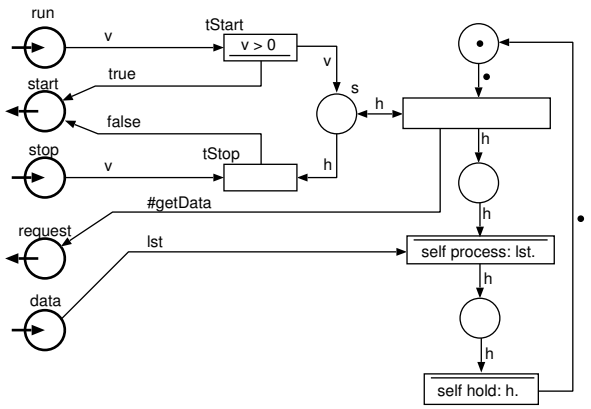


Figure 3. OOPN model of the component *Control*.

First, let us have a look at the component $Control$ (Figure 3). Input port $cn{\oplus}run$ expects a number h representing an interval of asking data from the component sn. Input port $cn{\oplus}stop$ expects any value—it only activates transition $tStop$, which suspends the loop. Both ports generate a command for coupled components through the output port $cn{\ominus}start$ (they put $true$ or $false$ to the mapped place $start$). The component cn asks for data by putting a symbol $\#getData$ to output port $cn{\ominus}request$ and waits for data (input port $cn{\oplus}data$). When the data are acquired, the method $process$: of the initial class c_0 is called and data are processed. Then, the loop waits for a given time unit h (the method $hold$:) and asks for data again.

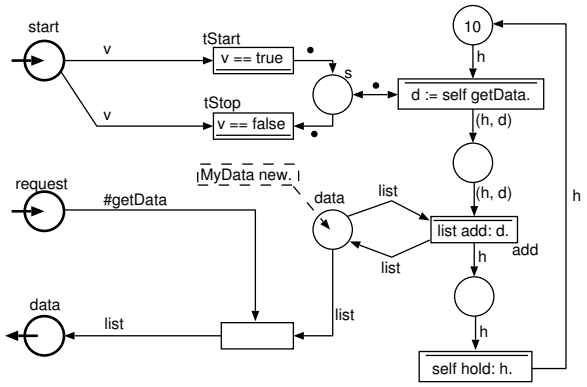


Figure 4. OOPN model of the component *Sensor*

Second, let us have a look at the component $Sensor$ (Figure 4). Input port $sn{\oplus}start$ expects values $true$ or $false$ activating transitions $tStart$ or $tStop$ that start or suspend the loop. The component receives a request for data by input port $sn{\oplus}request$ and puts data to the output port $sn{\ominus}data$.

The component sn acquires data in the loop, where the method $getData$ is called, the new data d is add by the transition add, and, finally, the loop waits for a given time unit h. The place $data$ contains an object (an instance of the class $MyData$) and the transition add simply adds new item by the method add:. Instance of the class $MyData$ is created and put into the place $data$ in the moment of object net instantiation (it is a place initialization, as shown in Figure 4).

D. Data Model Interface

So far, we did not care about actual data. Their form is not important for this paper (real numbers, integral numbers, etc.), only the way of data manipulation will be taken into account. It comes to this, that the data interface is important. Figure 5 shows identified interfaces and classes.

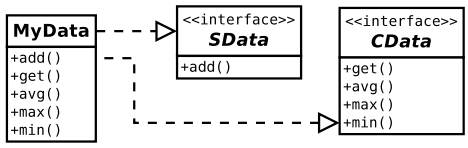


Figure 5. Classes and interfaces of data.

The component sn uses data storage by only way—adds a new item. So, the interface $SData$ containing the operation add: can be identified. Let us suppose, that the component cn needs following operations: get (getting an item), avg (average value), max (maximum value), and min (minimum value). These operations on data are performed within the method $process$:. Then, the interface $CData$ can be identified. The data storage (instance of the class $MyData$) used in the component sn (see Figure 4) has to implement the $SData$ interface. The storage object used in the component cn has to implement the $CData$ interface. Because both objects are identical (the object is carried through $sn{\ominus}data \xrightarrow{\mathcal{D}} cn{\oplus}data$), the class $MyData$ has to implement both interfaces, as shown in Figure 5.

V. Component interface consistency

The *System in Simulation* (SiS) concept assumes that components can be exchanged with no changes in the other components. In conjunction with the application framework [16], the component can be suspended, resumed, or changes any time during the system simulation. Therefore, it is necessary to be concerned with the problem of *component interface consistency*, in other words, the question whether component interfaces are compatible and whether its exchange is safe.

The component communication is provided by *data passing* [4]—the calling component (*client*) sends a data to the called component (*server*); the client does not need to wait for an answer. We will distinguish the structural aspect and the behavioral aspect of the component interface. The structural aspect is defined by ports and couplings. There is no problem to check if the components can be coupled or not. The behavioral aspect corresponds to the *concrete data* and their form.

A. Type constraints

Although the formalism of OOPN is pure object-based system and there is no need to define special kind of types instead of *class*, we will have special requirements to the set of types that can be checked:

- *a class* or *a subclass* – we need to check if the object is an instance of the class or its subclasses
- *an object interface* – we need to check if the object complies with the interface

Since we will check the type constraints, we have to define the term *type* in the context of the OOPN component system. CL_{env} is a set of classes from the product environment (the notation *product environment* is understood as the environment including language in which the application framework is implemented), $CL_{prim} \subset CL_{env}$ is a set of *primitive classes* (numbers, characters, and symbols), $I \subseteq \mathcal{P}(MSG)$ is a set of object interfaces, and ε represents a special kind of type meaning *unspecified type*. We define the interface I in the general way, as a set of operations that are independent from classes. The type is then $TYPE = CL_{PN} \cup CL_{env} \cup I \cup \{\varepsilon\}$.

Let T_P be a surjection $T_P : P \rightarrow \mathcal{P}(TYPE)$ assigning a set of types to a given place. The type of the place can be derived from the associations between classes, whereas there is no necessary to define only one type (and, thus, to allow all subtypes), but the set can be extended to next types. Implicitly,

 ISBN: 978-1-61208-438-1

each place has assigned the type ε. To discriminate between different levels of type constraints, we introduce following operators based on T_P:

- $\succeq$: $OBJ \times TYPE$ meaning the object is an instance of the class or derived classes, $\forall o \in OBJ : o \succeq \varepsilon$
- $\succ$: $OBJ \times I$, meaning the object complies with the interface, $\forall o \in OBJ : o \succ \varepsilon$

Let us continue with the example defined in Section IV-C. Figure 6–a shows type constraints defined on the input ports $stop$ and $data$ of the component $Control$ (cn). The port $stop$ requires any value of any type, so that the constraint is set to ε. The port $data$ requires objects of the class $MyData$, so that $\forall o$ in the place $data : o \succeq MyData$. The constraint will be written $\succeq \{MyData\}$.

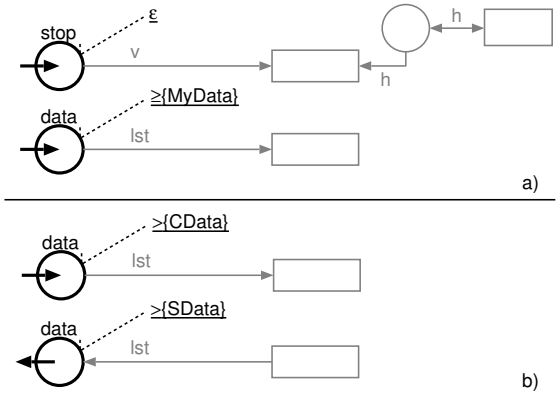


Figure 6. An example of type constraints.

Let us investigate a variant of interface usage—it is shown in Figure 6–b. There are depicted the input port $Control.data$ and the output port $Sensor.data$. Each of them operates with the different interface. $Control.data$ demands objects understanding methods defined by the interface $CData$, whereas $Sensor.data$ offers object understanding methods defined by the interface $SData$ (interfaces are discussed in Section IV-D).

B. Data constraints

Since the interface of the *common platform* is based on the principle of *data passing*, there will often be a request for constraints on data. First, let us define two auxiliary notions. Let $\mathcal{I}_G$ be a function $\mathcal{I}_G : TYPE \rightarrow \mathcal{P}(TYPE)$ assigning a set of generalized classes to the given class and $\mathcal{I}_S : TYPE \rightarrow \mathcal{P}(TYPE)$ be a function assigning a set of specialized classes to the given class. Then, $CL_{prim} = \mathcal{I}_S(Number) \cup \mathcal{I}_S(Character) \cup \mathcal{I}_S(Symbol)$. To discriminate between different levels of data constraints, we introduce following notions:

- *an enumeration* $\eta = \{e_1, e_2, \dots\}$, to check if the object o gets one of the listed values, $o \in \eta$; it can be used for symbols, numbers, or characters CL_{prim}
- *an interval* $\iota(i_1, i_2)$, to check if the object o gets a value from the interval, $o \in \iota(i_1, i_2)$; it can be used for numbers $\mathcal{I}(Number)$; there is a special value ω represents a maximal value or infinity

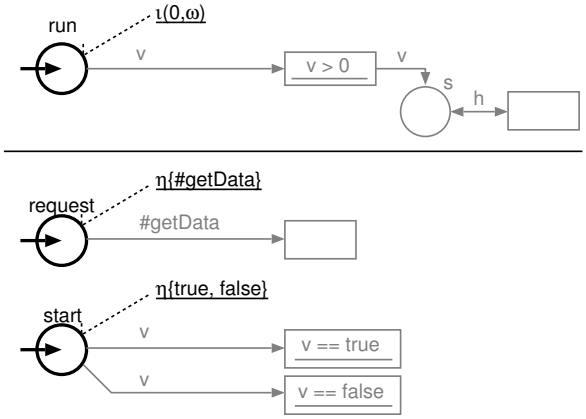


Figure 7. An example of data constraints.

Let us continue with the example defined in Section IV-C. Figure 7–a shows data constraints defined on the input port $cn \oplus run$. It requires a number from interval $\iota(0, \omega)$. Figure 7–b shows data constraints defined on the input port $sn \oplus (request, start)$. They require an enumeration of symbols or boolean values.

VI. Constraints Realization

Although the OOPN classes bring more intuitive modeling of behavior, they do not offer intrinsic definitions of *constraints* such as invariants or type checking. Nevertheless, there is very simple way how to define and test these conditions by means of OOPN [17]. Tests are generated by the application framework in accordance to required constraints defined on ports.

A. Type Constraints Testing

The test of *class constraints* is defined as $\theta_{\succeq}(p, ET) = \exists x \in p \wedge \nexists t \in ET : x \succeq t$, where x is an object in the place p and ET is a set of expected types. The test of *interface constraints* is defined as $\theta_{\succ}(p, ET) = \exists x \in p \wedge \nexists t \in ET : x \succ t$, where x is an object in the place p and ET is a set of expected types.

Both tests are implemented by negative predicates as shown in Figure 8. It follows the example defined in Section IV-C and shows two possibilities. First, the type constraint $\succeq \{MyData\}$ is defined for the input place $data$ of the component $Control$. This notion is equivalent to $\theta_{\succeq}(Control.data, \{MyData\})$. There is generated negative predicate $cTypeData$ and associated place ET containing a set of names of expected types. Names are stored in the form of symbols.

Firablity of the negative predicate is defined in two cases as follows. First, it is firable if there is no object in the associated place. Second, it is firable if the place is not empty and there is at least one object, which does not satisfy predicate conditions—on other words, the negative predicate finds all objects x that do not satisfy conditions. The condition is represented by arc expression t and calling special method $isKindOf$: t on the object x. The method is a part of object's *metaprotocol* and resolves in *true* or *false* depending

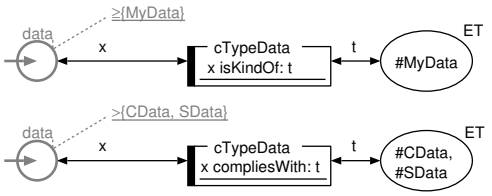


Figure 8. Type constraints realization.

on decision if the object is an instance of the class t (or its subclasses) or not. So, the predicate $cTypeData$ is *firable* if there is an object in the place $data$ and this object is not the instance of $MyData$.

Second possibility represents the type constraint $\succ \{CData, SData\}$ defined for the input place $data$ of the component $Control$. This notion is equivalent to $\theta_{\succ}(Control.data, \{CData, SData\})$. The constraint realization is the same as for $\succeq$ except that it uses the method $compliesWith$: instead of $isKindOf$:.

B. Data Constraints Testing

The tests of *data constraints* are implemented by negative predicates as shown in Figure 9. It follows the example defined in Section IV-C and shows two possibilities. First, the data constraint $\iota(0, \omega)$ is defined for the input place $start$ of the component $Control$. There is generated negative predicate $cDataStart$ having a condition corresponding to the defined interval. The predicate is firable if the condition is *not* satisfied.

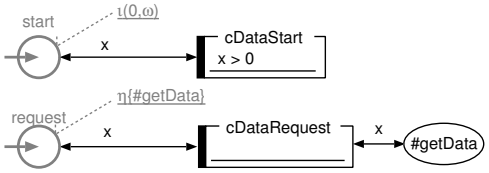


Figure 9. Data constraints realization.

Second, the data constraint $\eta\{\#getData\}$ is defined for the input place $request$ of the component $Sensor$. There is generated negative predicate $cDataRequest$ and associated place containing a set of expected symbols. The predicate is firable if there is found a symbol in the place $request$ that is not in the predefined set.

C. Exceptions

Constraints realizations presented in previous sections cannot be evaluated without calling them. Therefore, the new element of *exception* is introduced to the formalism of OOPN. The exception is demonstrated on the example of type constraint $\succeq \{MyData\}$ from Figure 8. The syntax is shown in Figure 10-a. The exception checks type constraint and if the constraint is not satisfied, it removes an object from the place $data$ and the associated *"any action"* is performed. The implementation is shown in Figure 10-b.

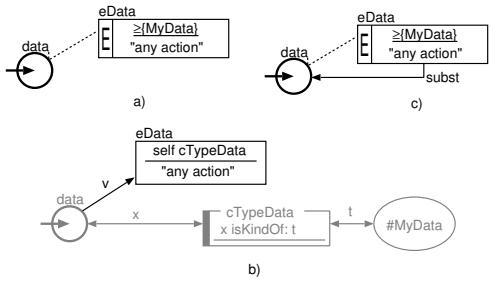


Figure 10. Invariants and testing conditions.

The exception may also have a side effect, e.g., it may offer substitute object and place it back to the place $data$ (shown in Figure 10-c).

D. Example of Component Changes

Let us continue with the example of common platform presented in Figure 2. The component $Sensors$ will be replaced by the component $Sensors2$ having the same structural interface, i.e., the same sets of input and output ports, as shown in Figure 11.

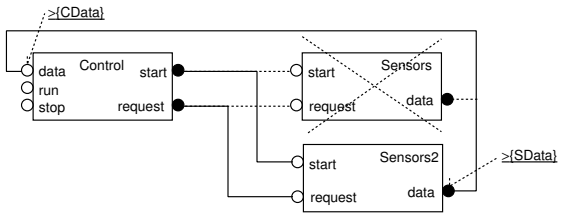


Figure 11. Common platform $\mathcal{M}_2$.

Let us suppose that the class $MyData$ has been changed to $MyData2$ containing only add: and get operations. Now, we will be only interested with $sn \ominus data \xrightarrow{\mathcal{D}} cn \oplus data$ coupling. The constraint $\succ \{SData\}$ is satisfied (operation add:), but the constraint $\succ \{CData\}$ is not satisfied (operations avg, max, and min are not present). Watching such incorrect changes, that do not have to be simply detected, allows to prevent systems from unexpected behavior.

VII. Conclusion and Future Work

The paper dealt with the concept of component platform based on DEVS and OOPN formalisms. It defined component interface and constraints above input and output ports. The interface is described by the means of OOPN places. Although they has assigned no type, for constraint testing it is possible to assign a set of types or constraints the objects have to satisfy. The concept of exception has been introduced to OOPN. Exceptions are a form of interface constraint testing. Incorrect changes done inside components do not have to easily be in evidence at the interface level. Constraints together with exceptions in languages that do not work with types allow to safe modification and changing component.

Future work will be aimed to a possibility to derive a set of types or constraints from the model analysis or simulation.

 ISBN: 978-1-61208-438-1

The component interface will be also generalized to other formalisms that can be incorporated into common DEVS platform.

ACKNOWLEDGMENT

This work has been supported by the internal BUT project FIT-S-14-2486 and the EU/Czech IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070).

REFERENCES

[1] R. Kočí and V. Janoušek, "Modeling and Simulation-Based Design Using Object-Oriented Petri Nets: A Case Study," in Proceeding of the International Workshop on Petri Nets and Software Engineering 2012, vol. 851. CEUR, 2012, pp. 253–266.

[2] M. Češka, V. Janoušek, and T. Vojnar, PNtalk — a computerized tool for Object oriented Petri nets modelling, ser. Lecture Notes in Computer Science. Springer Verlag, 1997, vol. 1333, pp. 591–610.

[3] R. Kočí and V. Janoušek, "Formal Models in Software Development and Deployment: A Case Study," International Journal on Advances in Software, vol. 7, no. 1, 2014, pp. 266–276.

[4] R. Kočí and V. Janoušek, "System Composition Using Petri Nets and DEVS Formalisms," in The Ninth International Conference on Software Engineering Advances. Xpert Publishing Services, 2014, pp. 309–315.

[5] S. Beydeda, M. Book, and V. Gruhn, Model-Driven Software Development. Springer-Verlag, 2005.

[6] M. Broy, J. Gruenbauer, D. Harel, and T. Hoare, Eds., Engineering Theories of Software Intensive Systems: Proceedings of the NATO Advanced Study Institute. Kluwer Academic Publishers, 2005.

[7] C. Raistrick, P. Francis, J. Wright, C. Carter, and I. Wilkie, Model Driven Architecture with Executable UML. Cambridge University Press, 2004.

[8] D. Cetinkaya, A. V. Dai, and M. D. Seck, "Model continuity in discrete event simulation: A framework for model-driven development of simulation models," ACM Transactions on Modeling and Computer Simulation, vol. 25, no. 3, 2015.

[9] S. U. Guan and S. S. Lim, "Modeling adaptable multimedia and self-modifying protocol execution," Future Gener. Comput. Syst., vol. 20, no. 1, 2004, pp. 123–143.

[10] M. Llorens and J. Oliver, "Structural and dynamic changes in concurrent systems: Reconfigurable petri nets," IEEE Transactions on Automation Science and Engineering, vol. 53, no. 9, 2004, pp. 1147–1158.

[11] J. Li, X. Dai, and Z. Meng, "Automatic reconfiguration of petri net controllers for reconfigurable manufactoring systems with an improved net rewriting system based approach," IEEE Transactions on Automation Science and Engineering, vol. 6, no. 1, 2009, pp. 156–167.

[12] N. Q. Wu and M. C. Zhou, "Intelligent token petri nets for modelling and control of reconfigurable automated manufactoring systems with dynamic changes," Transactions of the Institute of Measurement and Control, vol. 33, no. 1, 2011, pp. 9–29.

[13] K. Ohashi and K. G. Shin, "Model-based control for reconfigurable manufacturing systems," in Proc. of IEEE International Conference on Robotics and Automation, 2011, pp. 553–558.

[14] J. Liu and H. Darabi, "Control reconfiguration of discrete event systems controllers with partial observation," IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybertetics, vol. 34, no. 6, 2004, pp. 2262–2272.

[15] B. Zeigler, T. Kim, and H. Praehofer, Theory of Modeling and Simulation. Academic Press, Inc., London, 2000.

[16] R. Kočí and V. Janoušek, "The PNtalk System," 2015. [Online]. Available: http://perchta.fit.vutbr.cz/pntalk2k/

[17] R. Kočí and V. Janoušek, "Specification of UML Classes by Object Oriented Petri Nets," in ICSEA 2012, The Seventh International Conference on Software Engineering Advances. Xpert Publishing Services, 2012, pp. 361–366.

 ISBN: 978-1-61208-438-1

"Free" Innovation Environments: Lessons learned from the Software Factory Initiatives

Davide Taibi, Valentina Lenarduzzi
Free University of Bolzano-Bozen
Bolzano-Bozen, Italy
e-mail: {name.surname}@unibz.it

Muhammad Ovais Ahmad, Kari Liukkunen
University of Oulu
Oulu, Finland
e-mail: {name.surname}@oulu.fi

Ilaria Lunesu, Martina Matta
University of Cagliari
Cagliari, Italy
e-mail: {name.surname}@diee.unica.it

Fabian Fagerholm, Jürgen Münch
Department of Computer Science, University of Helsinki
Helsinki, Finland
e-mail: {name.surname}@cs.helsinki.fi

Sami Pietinen, Markku Tukiainen
University of Eastern Finland, School of Computing
Joensuu, Finland
e-mail: {name.surname}@uef.fi

Carlos Fernández-Sánchez, Juan Garbajosa
Technical University of Madrid (CITSEM & ETSISI)
Madrid, Spain
e-mail: carlos.fernandez@upm.es

Kari Systä
Tampere University of Technology
Tampere, Finland
e-mail: kari.systa@tut.fi

Abstract— **Entrepreneurs and Small and Medium Enterprises usually have issues on developing new prototypes, new ideas or testing new techniques. In order to help them, in the last years, academic Software Factories, a new concept of collaboration among universities and companies has been developed. Software Factories provide a unique environment for students and companies. Students benefit from the possibility of working in a real work environment learning how to apply the state of the art of the existing techniques and showing their skills to entrepreneurs. Companies benefit from the risk-free environment where they can develop new ideas, in a protected environment. Universities, finally benefit from this setup as a perfect environment for empirical studies in industrial-like environment. In this paper, we present the network of academic Software Factories in Europe, showing how Companies had already benefit from existing Software Factories and reporting success stories. The results of this paper can increase the network of the factories and help other universities and companies to set-up similar environment to boost the local economy.**

Keywords—Software Factory; Experience Report.

I. INTRODUCTION

Universities are perfect environments to exploit technological research for innovation. The biggest challenge to solve in universities is that they are nowadays rarely used by companies, and at the same time, universities are poorly oriented to give economic value while start-ups and Small and Medium Enterprises (SMEs) face new and tough challenges to survive in the market. Indeed, also big industries sometimes have difficulty being continuously innovative. In fact, ideas come out slowly and require a lot of effort to be implemented.

Fresh ideas, coming from the new digital native generation of developers, should encourage seniors to fresh thinking. From this aspect, the combination of university research, teaching and industry production would increase the value of skills of everyone and the development of innovation. Innovations lab of similar initiatives such as academic Software Factories (SF) [1] could contribute to fill this gap.

In SF, students and entrepreneurs collaborate together to develop a new idea or to apply existing techniques that couldn't be achieved by the entrepreneur itself without accessing to external resources. SF are university laboratories that emulate a real working environment, in which a given number of students, in the same location, work as a real team implementing a project for 7-11 weeks in a controlled environment with real customers and real deadlines. Entrepreneurs benefit from the new innovative ideas and the effort coming from students. Students have, in turn, a unique experience of working in an industry-like work environment getting in touch with the real business and a given number of credits. Moreover, students have the chance to present their skills to the entrepreneurs that can finally hire students partially trained on their technologies. The SF initiative is a fully bottom up initiative that cooperates on a voluntary basis without any funding framework, except for their enthusiasm and the common interest in getting excellent educational, and research results.

The goal of this work is to present the ecosystem of European SFs describing how best practices are shared between different software factories. Moreover, we aim at analyzing similarities and differences among SF in different countries (namely Finland, Italy and Spain), highlighting pros and cons for the different stakeholders. The results of the paper show how the use of SFs, as safe environment for developing new prototypes and products for start-ups or entrepreneurs, could represent a good practice and an important starting point for creating a connection between academic and working worlds.

The paper is structured as follows. After a first introductory section, we describe the SF concept in Section II. In Section III, we describe our international SFs, highlighting similarities

 ISBN: 978-1-61208-438-1

and differences. We report success stories in Section IV and finally we draw conclusions and future works in Section V.

TABLE I. SF BENEFITS

Academic Institutions	Companies	Students
• Perfect environment for empirical studies • Provide better training to their students. • Collaboration with industry • Environment for the development of research prototypes	• Environment to develop innovative ideas. • Environment to test new development tools or methodologies. • Opportunities for hiring new staff trained in the technologies that they use.	• First early contact with real-world projects • Develop transversal capabilities such as self-organized, responsibility, communication, etc. • Put into practice the theoretical concepts learned in the courses • Learn new techniques and technologies.

II. BACKGROUND

SF proposes itself as an infrastructure that supports research and education in software engineering and also entrepreneurship. In the Finnish editions, many collaborations with important organizations guided forward good results for customers and developers. SF is a safe and monitored environment that reproduces in a faithful manner the working team dynamics that develop a prototype or a software product for a customer, (SMEs) or a start-up. Since its first edition SF brings together three essential goals: Learn, Share, Grow. SF [1] represents also a shared educational platform for universities to hold courses where students are involved in a real-world project developing software in the same location or in different sites. SF relies on self-organization as its primary way of organizing the work [2].

It represents a unique platform in which a team of students develops software. SF projects are conducted in a manner that simulate as closely as possible, the reality of software development in the product development organization. We can then observe how SF could represent, despite its constraints and limitations, the operational core from which startups, entrepreneurs or SMEs could set up their own ideas allowing, at the same time, smart and brilliant students to make a unique practical experience learning by doing new methodologies and practices but also approaching the working world through the main door showing what they can do.

In addition, SF offers a way to learn new practices and technologies not only by reading from books but also by building a product. The results are achieved as a result of collaborative work of all team members, to improve their knowledge and skills getting in contact with people having different background and experience. As the students need to independently gain new knowledge and meet new people to create the product they get in touch with working reality and undertake new important collaborations. At the same time, SF are independent and open for collaboration with all kind of companies for entrepreneur or startups, the SF could represent a low cost environment in which they can set up new ideas and new projects in which create not only a prototype but meet partners and developers to be integrated in their own team having the advantage of a training period. The SF advantages exist also for researchers or academic members that would like to have the possibility to assist to the meeting of two worlds: work and study but in a monitored lab environment. This fosters the measures and observations to make research from a software perspective making measure about effort or software metrics and also from the educational perspective observing the student interactions, their learning behavior and their attitude in the creation of a new product. Table I summarizes the benefits of the SF environment.

III. THE SOFTWARE FACTORIES NETWORK

In this section, we describe the SF network in Europe presenting the different set-up and operational model.

A. The Helsinki SF (Finland)

SF at the University of Helsinki [1][8] has been organized since 2010. The factory deeply integrates the customer company into the development process. The customer provides a product owner who interacts directly with the student team during the project. The customer can range from local entrepreneurs to large enterprises and even to Open Source projects. For example, in spring 2015, the factory is participating in Facebook's Open Academy program, collaborating on two Open Source projects with universities worldwide. As a rule, five projects are arranged per year. The factory supports the projects with research-based insights for project management, methods, and pedagogy, and through full-time coaching of the teams.

B. The Bolzano-Bozen SF (Italy)

The factory [6][7] is organized by the Free University of Bolzano-Bozen. It is actively running once a year for 4 years, developing more than 10 projects. The participants are students from the first year of the Master program in Computer Science, third year of the Bachelor in Design and Education faculties, as well as local entrepreneurs. Project ideas come mainly from local entrepreneurs who are not affiliated to the university. The course runs during the summer semester for 11 weeks with a required effort of 200 hours per student. Students vote the projects to be developed based on their interests and skills. The most voted projects are developed during the SF. Students are then split in groups of 5-6 people and every group is assigned to a project. The entrepreneur who proposed the idea is required to be available at the SF at least once a week to support the students.

C. The Cagliari SF (Italy)

The factory has being running once a year for 3 years from 2012, developing a total of 4 projects. Participants are students from the Master program in Electronic Engineering, Telecommunication Engineering, Computer Science, PhD Course and local entrepreneurs. Projects come local entrepreneurs or ideas born for implementing applications to satisfy the needs of the research group. The course runs during the summer semester for 7-11 weeks with a required effort of 120-200 hours per student, with 4-8 people assigned to each project. The entrepreneur who proposed the ideas is required to be in class to support the students at least twice a week. The development is driven by an expert PhD student that plays the role of coach/coordinator. In order to replicate a real company

 ISBN: 978-1-61208-438-1

development environment, an open space is assigned to the team and the team members have to come twice or three times at week during the period.

D. The Joensuu SF

University of Eastern Finland's School of Computing established SF Joensuu in 2010 and is running 3-5 rounds per year. Teams consist of mostly master level computer science students with minimum target of 4 people. They are encouraged to participate two rounds, first round as software developer and later, second as team leader. Product ideas come from entrepreneurs and research groups with having target to produce new business opportunities or improve the world in general. SF team is supported by mentoring given by SF lead and students from previous rounds. Frequent interaction with customer is required in order to achieve release cycle of 1-2 weeks, preferably with face-to-face meetings at least at same interval and with other medium more frequently. Customers have been either starting entrepreneurs with just a good idea in their hands or already established companies from start -ups such as Epooq to big companies like CGI.

E. The Oulu SF (Finland)

The Oulu SF is established in 2012 to provide a realistic environment, which improves the students' learning experience by providing them with insights into the conduct of real-life software projects with close customer involvement, intensive teamwork, and the use of modern software development tools and processes [5][7]. As a platform, it serves multiple purposes. It is a test bed for software engineering ideas and a source for original basic scientific software development research. Oulu SF runs twice a year and it has completed more than 8 projects since 2012. The participating students are from first and second year of master's degree in information processing science. The project tasks come from the local software companies and or research projects. Each project involves a minimum of four members. The students are encouraged to tackle management and resource planning issues pertaining to large teams. Each project team is assigned a project supervisor who provides the team with technical and non-technical guidance. The supervisor is also responsible for monitoring and assessing the team throughout the course of the project.

F. The Madrid SF (Spain)

Madrid factory has been operating since 2011. The factory was a joint set-up between the Technical University of Madrid (UPM) and Indra Software Labs, a subsidiary of Indra, a Spanish global company. Actually, two software factories were set up, one in UPM and one in Indra Software Labs, to run joint projects. As an educational setting, students that participate are from a Degree in Software Engineering and Masters Program on Computational Science and Technology. Most of the projects are closely related to tasks of European or National Research projects, very often collaborative projects between industry and UPM. Different kind of projects have been performed over these years, and more than 10 projects: five of them were distributed projects, in which up to three nodes from different countries were involved, e.g., Helsinki (Finland) and Bolzano (Italy). The Factory recruits students usually in November and February. The recruitment process includes an interview to applicants with a given number of questions. Even when some these questions are technical, other skills are also searched, also considering the kind of project that the student would like to be involved, and the positions available for each project. Students usually work in teams, from 3 to 6 students. Students perform slots of 140 hours, during generally 8 weeks, and up to a maximum of 3 of these slots.

IV. SIMILAR ACTIVITIES IN TAMPERE (FINLAND)

A project course with many similarities to SF has been run at Tampere University of Technology from year 1991. Already during the first years, we received project ideas from companies, and this collaboration has been a key component of the of project course. Currently, the project ideas come from the companies and companies give constant feedback about progress of the project and produced software. These companies essentially play a role of customer for the student team. Since the course has a long history in Tampere, many managers in the surrounding companies have participated in the course in the past, they now have a high motivation to collaborate as a customer. Key role of the companies is seen valuable for both students since it gives both parties an opportunity to network.

In addition to independent project work, the course also includes some lectures that help students in management of the project. There are also lectures about IRR, legal and business aspects. The volume of the project course has been varying over the years. During academic year 2014-2015 there were 10 project teams with 5-8 members each. The number of hours spent per student is in the range of 130 – 260 hours. The course is run yearly starting September and ending in February to an end seminar and celebration.

Since 2008, Demola [9], has also been another option in Tampere. Demola focuses on innovation projects, where students are asked to further develop ideas given by surrounding companies and public institutes. Demola is hosted by Hermia (a business development company) and three universities, Tampere University of Technology, University of Tampere and Tampere University of Applied Sciences, participate in Demola. The project teams are cross-disciplinary and consist, e.g., of engineering, business, and design students from participating universities. Since Demola projects concentrate in innovation and further development of the idea, the process includes value creation workshops and pitching events. Furthermore, Demola shares promises with Protomo [10], which is a development community for new businesses and start-ups

Regular Demola projects run twice a year: once in the fall and once during the spring. The volume of 20-25 projects per season typically run by groups of 4-5 students. The main difference between Software Engineering Project Course and Demola is that the former concentrates in professional development projects while the latter concentrates in idea development and innovation.

 ISBN: 978-1-61208-438-1

V. SOFTWARE FACTORY SUCCESS STORIES

Software factories actively supported local entrepreneurs. Here, we report on some of the most important success stories of our factories.

A. Innovative Video Calling Service

Between September 2012 and December 2013 the SF Helsinki conducted three projects together with Tellybean Ltd., a small Finnish startup [3]. The vision of this startup was to deliver a life-like video calling experience for specific customer segments such as elderly persons. The overall goal of the collaboration was to conduct build-measure-learn loops to validate critical assumptions underlying the business model and the technical solutions in order to rapidly learn if the chosen strategy needs adjustments or can be persevered. The first project focused on the development of appropriate analytics for measuring the performance of the video service so that business-critical information can be gathered and analyzed. In addition, technical feasibility aspects were analyzed. The second project mainly focused on validating technical assumptions. In particular, the company wanted to understand the scalability and robustness of the proposed system architecture, technical weaknesses of the system, and the company wanted to identify alternative options for the system architecture. The project resulted in a significantly better understanding of the limitations and future development options. The third project helped the company to better deploy functionality in a continuous way.

Overall, the prototypes that were created and used in the projects served as so-called minimum viable products to quickly validate business-critical assumptions and helped the startup to accelerate learning about their ideas. In the meantime, the startup got significant funding. Now, Tellybean partners with major service providers. The SF Helsinki benefited well from these projects by learning how to organize industry-academia collaborations in order to test business-critical assumptions.

B. Memoree

Memoree was a SF project at Bolzano in spring 2013. It was based on a business idea from a local entrepreneur who needed to develop a prototype to prove his idea. The initially intended software solution would pack personal photos, videos and audios into a memory package and shared it among friends. The project lasted 11 weeks. In total, 14 students were involved in the Memoree project. The majority came from the Computer Science faculty. Two designers were involved at the later stage of the project. The entrepreneur played the customer role for the project and made himself available all through the SF session. The Memoree SF project was very useful for the local entrepreneur to understand what are the crucial features of Memoree, and what should be skipped. SF also helped him to decide what could be the core component of the application. The developed prototype was very different than the initial idea that he had. The students were not just implementing the prototype, they were contributing to the understanding of the need the startup intended to meet, and the clarification of the vision that drives the startup. After the SF session, the Memoree idea became more concrete. It is positioned as a mobile application that is developed for automatic creation of videos based on different contents (photos, songs, etc.). The app provides content privacy, and creates videos automatically by taking songs and photos as an input. This application is composed of two modules: content management (photos) and video creation. The intended customers are iOS users. The entrepreneur team was expanded from a single person to five founders (two economics, one finance, one graphic designers and one computer scientist). The actual development started in May 2014.

C. Medygo

Different from the Memoree case, when a founder of Medygo approached the Bolzano SF in Spring 2014, the business idea has already been validated initially with potential customers, and a prototype was developed already. It is a mobile application that is developed with motive “health on go”. It is mainly developed for people to solve their health problems during traveling and staying abroad. The main purpose of developing this application was to prevent travelers from the hustle when they travel and become sick during their journey. This mobile app converts medicine, what they take in their own country, to what they should take in another country. This app is initially developed for android users. There were four founders (two businessmen, two pharmacists). It’s been one year since they have been working on this idea before they contacted the Bolzano SF. The actual development started in November 2013 by adding another team member as a developer. The initial version was launched in January 2014.

One of the benefits of working in SF was that, recalled by one founder, was the iterative approach the SF adopted. There were always some deadlines, and the team had to finish on time. During the SF session, the Medygo team set milestones e.g., two-week idea validation, two-week date collection, four-week development and so on. At the end of the 10th week, their prototype was ready. The team was also facilitated by the SF tutors to handle pressure, and to meet deadlines. In addition, the SF students worked on the project became potential hire for the startup company due to the intimate knowledge they obtained through working on the project at the SF.

D. Matchall2

The Matchall2 project was proposed by a local entrepreneur that played the customer role during the development period, with the aim to build a plugin for categorizing personal multimedia content gathered from famous social networks such as Facebook, Youtube and Flickr. Matchall2 created a personal communication engine based on innovative principles and functionalities, with a web implementation and diffusion strategy. The final prototype, thanks to an idea of some developers, was represented by a bookmarklet that allowed one to easily classify and categorize personal content, such as pictures and videos, in a customized manner using tags. The focus of the project was to implement the same application for different social networks. This SF started in early March of 2013 and held 11 weeks involving 8 specialized students with rich and different skills and backgrounds.

In order to take advantage of developer's skill diversity, the development was organized considering pairs in which an expert developer supported a less experienced student.

 ISBN: 978-1-61208-438-1

In this edition, many technologies and new abilities are used and learned to obtain the maximum result. The meetings with the entrepreneur allowed to stay in the right edges of time and specifics. The particularity of this case is that people also when they don't know each other thanks to the fact they have the same aim, strive to implement a success product, behaving as a family helping each others to solve problems or achieve the same objectives.

E. SERTS (Software Engineering Research Tool Suite)

During the SF of 2013 edition, the project Software Engineering Research Tool Suite (SERTS) has been developed by a team of 6 students. The aim was the implementation of a semi-automatic tool able to semplify the analysis of data collected in software repositories such as Bugzilla, CVS, SVN, Git, and Jira. The development period lasted eight weeks, from September 2013 to November 2013 by a team composed by six developers: a PostDocstudent, four PhD students and one undergraduated. In this specific project, a medium knowledge of software development was required. Each component of the team had different tasks, chosen according to their skills. One of the PhD students with a strong knowledge of the technologies involved into the project, played the role of team coordinator/coach. The used development process was Scrum with iterations of two weeks.

The figure of the customer was very significant. Every two weeks he monitored the work of the students observing the progress of the project through spikes. Due to its constant presence, it was possible to build a prototype inherent to its requirements.

F. FREI MARKT SÜDTIROL

In 2014, Oulu SF and Bolzano SF start a collaborative time banking project “FREI MARKT SÜDTIROL”. An Italian entrepreneur was sending requests for his project to Oulu SF team. The project idea was to provide a common single plate form for existing time banking systems in South Tyrol and other near cities.

Project aim was to provide a fresh new time bank-community system which cover various parts of society and particularly for those people who are strongly hit by the ongoing socio-economic crisis including young unemployed, working-poor and immigrants. A system was developed which allows users to create their personal profiles, look for jobs & products, post jobs & products, apply for jobs & product and give feedbacks. In addition, a SMS platform will facilitate the new member registration process, modification and verification of time-checks (BiX) when the people are not familiar with the Internet. Project team consists of eight students; in which four students working from Oulu SF and four from Bolzano SF. Both teams were having mentors to help agile and lean concepts in the project. The use of Kanban method and JIRA was mandatory for Oulu SF students while the Bolzano SF students were not following any specific methodology or practices. Both teams use and get experience with Rise Editor, Myeclipse, Apache Tomcat, PostgreSql, Dreamweaver, and GitHub in the project. In first two weeks, students attend mandatory lectures and exercises in SF. During weeks 3 to 5 literature was studied related to the project idea, working methodology and preliminary project plan were drafted. Then design and actual development related tasks were carried out within weeks 6 to 12. After every two week the teams deliver batch of minimum viable product to customer. The project demo was given in Bolzano University which was appreciated and covered by local press.

G. Google Glass for Traffic Warden

One of the latest Demola examples is a project where five students got the idea from a local SW company Vincit but also collaborated with City of Tampere. The project integrated automatic recognition register plates to Google Glasses. In this project, the student group developed the first commercial smart glass application in Finland. With this application the traffic warden is able to see right away if the parking ticket has been paid or not. The city of Tampere is piloting this system in spring time 2015 [14].

Development of this system may not have been possible with traditional processes where software companies and public authority as customer should recognize the idea first and then have detailed enough specification. In this case, the student group approached the idea as a start-up by trying and doing. The system project received also a fair amount of publicity in Finland.

H. Optimeter

The Optimeter project was developed in 2012 by the Madrid SF (Technical University of Madrid and Indra Software Labs) and the Helsinki SF (University of Helsinki). The Optimeter project (in practice there were two projects inside the SF, Optimeter I and the subsequent Optimeter II) had as goal to implement some use cases about data acquisition in intelligent power networks, usually known as power smart grids. The objective was to build a benchmark to validate massive raw data coming from sensors and smart meters. The benchmark was created using Apache Hadoop and Oracle NoSQL Database to provide distributed processing and storage capabilities to the system. Optimeter I and II were traversal to two European projects under the ITEA2 Programe: IMPONET (Intelligent Monitoring of POwer NET [11], 127 man years) and NEMO&CODED (Networked Monitoring & Control Diagnostic for Electrical Distribution [12], 112 man years, and a third Spanish project called ENERGOS (Technologies for automated and intelligent management of power distribution networks of the future [13], with a budget of 24.3 million euros). The project was developed using agile practices, and more concretely following the Scrum methodology. Optimeter was an excellent framework to set up a collaboration activity between three Software Factories (UPM and ISL in Madrid, and UH in Helsinki).

One lesson learned is that the training that the students can get in such environment is very useful but straining. Students were under the same pressure that the industrial development team during the weeks the project took place. But at the end, the background, skills and experience were very much welcome by the students.

From the point of view of the industry, they could develop the software that they needed, experimenting the usage of agile methodologies in a distributed development environment. Also they use the project to test some development tools that they were not using until that moment.

 ISBN: 978-1-61208-438-1

I. HavuSport

In 2013, School of Computing in Joensuu was contacted by two hockey coaches with an idea that the junior coaching should be supported by an electronic system, usable with different end user devices. Mobile device is the device of the day that younger people easily relate to. The system should support all major activities of a hockey team from messaging and timetabling to performance statistics while getting rid of excel-sheets, paper and pen. They could not find a proper existing system, so they decided to have it build. SF Joensuu built a web-based system with mobile applications coming aside during two project rounds. Team size was 8 people, but the project needed to be scoped very well because of the high amount of required features and the fact that there was a lot to learn in a short period of time.

Team felt a real business pressure to deliver, a feature that is build inherently to SF, succeeding to achieve the target in time. They felt proud that their hard work paid off. Havusoft Company Ltd. was formed around the product and now Havusoft is planning to extend the software system for other sports activities too and there is great interest in the market to use the system. The major role of SF concept in this process was to enable starting entrepreneurs to push their idea forward and show to the world that they are serious with their endeavor.

VI. CONCLUSION

In this paper, we present the academic Software Factories (SF) in Europe, describing how they can help the local economy by means of the collaboration among academia, entrepreneurs and SMEs.

Our goal is to report on our SFs and similar initiatives, presenting success stories.

SF provide an unique environment where entrepreneurs can explore new ideas, develop new prototypes or apply new techniques and students can study and work in a setting that replicate, as much as possible, a real work environment. Moreover, students have the opportunity to show their skills to entrepreneurs and entrepreneurs can find new developers easily, based on a direct knowledge of the students itself.

The network of SF in Europe, shared among Finland, Italy and Spain is composed by several University that serves hundreds entrepreneurs. SF collaborates with some shared projects, working for the same project collaboratively, such as the project described in the success story “Frei Markt Sudtirol”. We reported several success stories, such as the Google Glass for Traffic Warden, Memoree, Optimeter and others, showing how the different stakeholders benefit of the SF environment, from an entrepreneurial, didactical, and research points of view.

A new Software Factory has recently been established at Montana State University in Bozeman, MT. A first project with a company from the financial sector has started and relationships with entrepreneur communities such as Blackstone Launchpad have been established.

In the future we plan to further expand the community extending the number of partners’ universities and increasing the number of shared projects and involving industries in the project selection and execution.

Finally, future works include the analysis of different development approaches adopted in the SFs, so as to understand if success stories are caused by the agile approaches or for other reasons.

ACKNOWLEDGMENTS

The authors would like to thank the companies and their employees for participating to this research. This research has been carried out in Digile Need for Speed and Digital Services programs, and it has been partially funded by Tekes (the Finnish Funding Agency for Technology and Innovation), the Italian Regione Autonoma della Sardegna (RAS), Regional Law No. 7-2007, project CRP-17938 LEAN 2.0, the Spanish projects iSSF (i-Smart-Software-Factory) IPT-430000-2010-38, INNOSEP TIN2009-13849, IMPONET ITEA 2 09030 TSI-020400-2010-103, NEMO-CODED ITEA2 08022 NEMO CODED IDI-20110864, and ENERGOS CEN-20091048.

We also thank Ville Korpiluoto from Demola (Tampere) and Xiaofeng Wang (Free University of Bolzano) for reviewing and supporting this paper.

REFERENCES

[1] P. Abrahamsso, P. Kettunen and F. Fagerholm, ”The set-up of a software engineering research infrastructure of the 2010s.” In Proceedings of the 11th International Conference on Product Focused Software ACM. pp. 112-114, 2014.

[2] X. Wang, I. Lunesu, J. Rikkila, M. Matta and P. Abrahamsson, ”Self-organized Learning in Software Factory: Experiences and Lessons Learned”. In Agile Processes in Software Engineering and Extreme Programming. pp. 126-142, 2014.

[3] F. Fagerholm, A. Sanchez Guinea, H. Mäenpää and J. Münch, “Building Blocks for Continuous Experimentation”. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (RCoSE 2014), Hyderabad, India,. pp 26-35, June 2014.

[4] M.O. Ahmad, K. Liukkunen and J. Markkula, J., “Student perceptions and attitudes towards the software factory as a learning environment”. IEEE Conference on Global Engineering Education. Istanbul, Turkey. pp 422 – 428, 2014.

[5] M.O. Ahmad, J. Markkula and M. Oivo, “Kanban for software engineering teaching in Software Factory learning environment”. World Transactions on Engineering and Technology Education (WIETE), Vol.12, No.3, pp 338-343, 2014.

[6] V. Lenarduzzi, I. Lunesu, M. Matta, and D. Taibi, “Functional Size Measures and Effort Estimation in Agile Development: a Replicated Study”, in XP2015, Helsinki, Finland 2015

[7] Bolzano-Bozen Software Factory, http://www.newsoftwarefactory.org (Accessed: June 2015).

[8] Helsinki Software Factory, http://www.softwarefactory.cc. (Accessed: June 2015).

[9] Demola. www.demola.fi. (Accessed: June 2015).

[10] www.protomo.fi. (Accessed: June 2015).

[11] IMPONET, https://itea3.org/project/imponet.html (Accessed: June 2015).

[12] NEMO&CODED, https://itea3.org/project/nemo-coded.html (Accessed: June 2015).

[13] ENERGOS, http://innovationenergy.org/energos/ (Accessed: June 2015).

[14] Google Glass for Traffic Warden. http://googleglassfortrafficwarden.blogspot.fi (Accessed: March 2015).

 ISBN: 978-1-61208-438-1

Performance Exploring Using Model Checking

A Case Study of Hard Disk Drive Cache Function

Takehiko Nagano[1,3], Kazuyoshi Serizawa[1], Nobukazu Yoshioka[2], Yasuyuki Tahara[3] and Akihiko Ohsuga[3]
[1]Research & Development Group, Hitachi, Ltd., Yokohama, Japan
[2]GRACE Center, National Institute of Informatics, Tokyo, Japan
[3]Graduate School of Information Systems, University of Electro-Communications, Chofu, Japan
e-mail: {takehiko.nagano.nr, kazuyoshi.serizawa.fz}@hitachi.com, nobukazu@nii.ac.jp, {tahara, ohsuga}@is.uec.ac.jp

***Abstract*—To avoid performance problems (e.g., execution delay), model-based development represented by model checking is used to improve performance quality. However, not so many studies have applied the model checking of performance to actual product development. Specifically, model checking has not been applied to performance exploring, so it is hard to say how effective model checking is. Furthermore, creating a new model for performance verification in addition to the usual development process greatly burdens developers. To reduce this burden, man hours for performance verification modeling must also be reduced. Accordingly, we embedded parameter deployment code to create a performance verification model and achieved performance exploration to ease performance optimization. Also, we developed a performance verification modeling method reusing existing product code to reduce modeling costs (man hours). In this paper, we report a case study in which the proposed method was applied to a Hard Disk Drive (HDD) cache emulation program. According to the results, the minimum cache capacity required processing was completed within the target time. We also show that 57.89% of cache emulation program codes were reused to create the new performance verification model. These results validated the proposed method.**

Keywords-performance; model checking; embedded system.

I. INTRODUCTION

Embedded computer systems acquire more advanced features and become more complicated every year, so the lines of code also increase. Therefore, the parameters that control the system increase, the combinations of the processing that attains performance become huge, and the performance prediction and exploring of the system are difficult. For example, in the database software case, although the tuning parameter is prepared, performance optimization is not carried out for each product. Thus, system engineers need to do performance tuning using the above parameter before product release. Therefore, the tuning documents and tools are prepared by the software vender [11]. Moreover, system engineers need to explore system performance including hardware controlled by software and other software packages. However, if performance tuning is not finished by the release deadline and products are released while still having performance problems, we may suffer damaged customer relations, business failures, income loss, additional project resources, reduced competitiveness, and project failure [2]. Complicated product exploring is difficult to fit in to the limited time of a product's release schedule. Compuware reported that 20% of computer systems have performance problems (e.g., execution delay) [13].

To solve these problems, usually two approaches have been taken. One is carrying out performance prediction and design at early phase of system development. The other is verifying, analyzing, and solving the performance problems at later phase of system development [1][2].

Specifically, at early phase of system development, we carry out system performance prediction using a mathematical model represented by queuing theory [3][4] and performance verification of an algorithm using model checking represented by UPPAAL [6][16][17]. At later phase of system development, we carry out implementation based on a design using the above techniques and performance evaluation, analysis, tuning, and redesign using test results [2]. These techniques have achieved positive results. However, it is difficult to evaluate and analyze performance comprehensively. Because, the parameters that control the system increase, and the combinations of the processing that attains performance become huge. In this paper, we focus on model checking from the viewpoint of comprehension. And we apply it to performance exploring.

The case studies of using model checking are reported [6], [7][8]. However, not so many studies have applied the model checking of performance to actual product development [16][17]. Specifically, model checking has not been applied to performance exploring, so it is hard to say how effective model checking is. Furthermore, creating a new model for performance verification in addition to the usual development is a big burden for developers. To reduce this burden, man hours for performance verification modeling must also be reduced.

In this paper, we propose the following two methods:

1) An easy performance exploring method embedding parameter deployment code used to create performance verification model;

2) A performance verification modeling method reusing existing product code to reduce modeling costs (man hours).

By method 1), performance exploring realizes a comprehensive verification mechanism of model checking. Moreover, by method 2), the C code embedded function of PROMELA is used for performance verification modeling [20]. Specifically, costs are reduced by using the actual product C code instead of new modeling by PROMELA.

Moreover, we report a case study in which the proposed method was applied to a cache emulation program.

 ISBN: 978-1-61208-438-1

In Section 2, we describe a performance problem and objective. In Section 3, we explain our proposed method. In Section 4, we present about our target, a HDD. Specifically, we present a cache emulation program and analysis results of its application. In Section 5, we discuss the effect of the proposed method. In Section 6, we detail our conclusions and future work.

II. PROBLEM AND OBJECTIVE

A. Performance problem and research scope

A purpose of this paper is to solve the execution delay problem of the embedded computer system. We assume that all programs are implemented in C language in this paper, because C is a major programming language in embedded systems. Particularly, a target of this paper is an embedded system in that software controls hardware, such as a storage system, a car engine controller and so on.

B. Related works

To solve these problems, many techniques have been proposed and applied. To overcome system performance problems, two approaches have been taken. One is carrying out performance prediction and design at early phase of system development. The other is verifying, analyzing, and solving the performance problem at later phase of software development. Below, examples of these approaches are presented.

1) Countermeasures against performance problems at early phase of system development

At early phase of system development, we carry out system performance prediction and performance verification of an algorithm. Performance prediction uses a mathematical model, typically queuing theory. Queuing theory has been applied in various fields, and many results have been reported [3][4]. Moreover, an example using the Markov model for the performance prediction model has also been reported [5].

Next, the prediction and verification using a design model are described. The modeling method consists of a mathematical model and a programmatic model. In the mathematical model, the model is created using timed-automata [9], Petri net [18], and so on. In the programmatic model, the model is created using UML extended by MARTE [1]. The performance design and verification using model checking is included here. UPPAAL using timed automata is a widely used model checking tool in this domain [6][16][17]. For example, UPPAAL is applied to time constraint verification of Audio/Visual protocol [6]. There are also other models checking tools like PRISM that can verify a statistical model [7].

2) Countermeasures against performance problems at later phase of system development

At later phase of system development, we carry out two main performance improvement measures. One is a performance analysis test of a developed system to evaluate whether the target performance is achieved. The other is performance tuning to analyze test results. After that, the system is redesigned, parameters are reconfigured, etc. [1][2]. These techniques have been applied to actual systems, and designs for next generation products have been reported [15]. Moreover, our company also applies these measures in many product developments. Furthermore, documents and tools needed to master a software package are prepared by the software vender [11].

C. Problems to solve

The countermeasure described in Section 2-B is implemented to prevent performance problems. And, these techniques have achieved positive results. However, it is difficult to evaluate and analyze performance comprehensively. Because, the parameters that control the system increase, and the combinations of the processing that attains performance become huge. In this paper, we focus on model checking from the viewpoint of comprehension. Also, we apply it to performance exploring.

Not so many studies have applied the model checking of performance to actual product development. Specifically, model checking has not been applied to performance exploring, so it is hard to say how effective model checking is. Moreover creating a new model for performance verification in addition to the usual development greatly burdens developers. Furthermore, to reuse old product code, it is necessary to create a performance verification model that also includes the past code. This recurrent work also becomes a big burden. To reduce the above burdens, man hours for performance verification modeling must also be reduced.

As a result of the above issues, the problem to solve is as follows.

Problem to solve: Enable performance exploring of complicated systems with advanced features.

To solve the above problem by model checking, we first do the following.

- Establish a method for applying model checking to performance exploring
- Develop an efficient performance modeling method

III. PERFORMANCE EXPLORING USING PARAMETER DEVELOPMENT AND PERFORMANCE VERIFICATION MODELING REUSING PRODUCT CODE

There are various types of performance, such as execution time and throughput. In this paper, we define execution time as performance.

A. Outline of proposed method

Many modeling languages exist for design and verification. Modeling languages for design include UML, and modeling languages for verification include model checking such as PROMELA [20]. Furthermore, there are two types of language for verification. One is for functional verification such as PROMELA, and the other is for verification for real time systems such as UPPAAL [6]. In this paper, our target is a modeling language for functional verification such as

 ISBN: 978-1-61208-438-1

PROMELA. Because model checking is used, comprehensive verification is attained. Additionally, by applying model checking, performance exploring is achieved. From the above, we propose the following two methods.

1) Easy performance exploring using parameter deployment code
2) Performance verification modeling reusing product code

By method 1), we can apply model checking to performance exploring. Performance exploring is realized using the comprehensive verification mechanism of model checking. Moreover, by method 2), we can develop an efficient performance modeling method. We use the C code embedded function of PROMELA for performance verification modeling. Specifically, costs are reduced by using actual product C code instead of new modeling by PROMELA. Here, *FeaVer*, which generates the PROMELA model from the C code, exists as related research. However, *FeaVer* is not a performance verification model but only a functional verification model [9].

Moreover, we explain how to verify HDD performance using PROMELA/SPIN not aimed at real-time verification, unlike UPPAAL.

B. Performance exploring using parameter deployment code

In case that there are some parameters affecting to system performance, to find a set of the parameters to achieve required performance, performance exploring of the parameters needed to repeat until adequate set was found.
We propose a parameter exploring method for performance to let a model checker, like SPIN. For example, in selecting cache size, we want to choose the smallest cache that satisfies the target performance. In this case, after the cache size is changed, many tests must be performed and results evaluated. When a tester uses a simulation program, the program evaluates by creating a script as shown in Figure 1. In Figure 1, the caches sizes in the second line (4, 8, 16, 32, and 64MB) are inputted to the cache_simlator program, and all patterns are executed to calculate execution time.

```
1 #!/bin/sh
2 for CACHE in 4 8 16 32 64
3 do
4./cache_simulator workload_cmd_data.csv $CACHE > result$CACHE.txt
5 done
```

Figure 1. Wrapping program

By using a model checking technique, SPIN deploys parameters for exploring. Furthermore, the machine was checked to see whether verification conditions were satisfied. To evaluate cache size, as shown in Figure 2, all cache sizes that can be taken in "if" sentences must be described. By this description, the verification machine (SPIN) verifies by exploring using all parameters. Thereby, to create a script as shown in Figure 1, performance test using an actual machine, analysis of the result log, etc. become unnecessary, and performance exploring efficiency improves.

```
1 if
2 :: CacheSize_MB = 4
3 :: CacheSize_MB = 8
4 :: CacheSize_MB = 16
5 :: CacheSize_MB = 32
6 :: CacheSize_MB = 64
7 fi;
```

Figure 2. Parameter deployment sample

C. Performance verification modeling reusing product code

1) Reuse of whole processing

The part that does not contain the conditional branch that influences performance reuses the original C code. The only thing necessary is to surround the function of C language with the c_code{}. An example is shown in Figure 3. In Figure 3, the function sorts a segment's structure by time using qsort of libc. To apply this technique, it is necessary to check whether the target function is processed atomically. This is because the inside of the processing surrounded by c_code{} is processed atomically by SPIN.

```
1 c_code{
2  //compare function
3  int comp_segment(const void *seg1,const void *seg2)
4  {
5    int Time1,Time2;
6     SegmentUnit *Unit1 = *(SegmentUnit **)seg1;
7     SegmentUnit *Unit2 = *(SegmentUnit **)seg2;
8
9     Time1 = Unit1->Time;
10     Time2 = Unit2->Time;
11
12     return Time1 - Time2;
13  }
14}
```

Figure 3. Example of call function writing by C code

2) Modeling of the part containing conditional branch that influences performance

In this subsection, we describe modeling the part containing the conditional branch that influences performance. In the proposed method, the conditional branch (if, while, etc.), which has influence on performance need to be converted to conditional branch of PROMELA, and about expression of the condition, the original C code need to be surrounded with the c_expr{}.

Figure 4 shows the original C code of the conditional branch, and Figure 5 shows an example in which it is PROMELA-ized. The control structure of C language can be mostly used by PROMELA: "if" sentence, "while" sentence, etc. Thus, we use it as shown in Figure 5.

```
1  if(LRUDumpTime ==0){
2       SystemTime += TimeInterval;
3 }else{
4       SystemTime += LRUDumpTime;
5          LRUDumpTime = 0; }
```

Figure 4. Example of original C code

 ISBN: 978-1-61208-438-1

```
1 if
2 ::c_expr{ LRUDumpTime == 0} ->
3   c_code{
4       Pcache_main->SystemTime += TimeInterval;
5   };
6 ::else ->
7   c_code{
8       Pcache_main->SystemTime = += LRUDumpTime;
9       LRUDumpTime = 0;
10  };
11 fi;
```

Figure 5. Example of PROMELA model

For example, when the "if" sentence shown in Figure 4 is written by PROMELA, the whole code is surrounded by "if" and "fi" like in the first and eleventh lines in Figure. 5. Conditional sentences are written like the second and sixth lines. Moreover, we need the cross-reference of the variable declared within the model of the PROMELA portion and the variable declared in the C code portion. In this paper, the variable declared within the model of PROMELA is updated by the C code side and then used for PROMELA model control. For example, the fourth line in Figure 5 is equivalent to this processing. In this case, the variable "SystemTime" declared by PROMELA is updated by the C code side. If SPIN can be distinguished in the variable of the PROMELA process, SPIN cannot be renewed. In this case, "Pcache_main" describes the PROMELA process information. P represents a process, and cache_main represents the process name. By following this notation, SPIN can execute a name resolution so that an applicable variable can be referred to.

IV. HARD DISK DRIVE CACHE EMULATION PROGLAM AND ANALYSIS RESULT

In this section, we describe the analysis results for applying the technique of performance verification and exploring described in Section 3 to a HDD cache emulation program. Moreover, we describe the application of the technique using the analysis results.

Therefore, first, we describe the HDD cache emulation program used this time. Next, we describe the analysis results of the cache emulation program. Furthermore, we describe the modeling of reusing actual cache emulation program code. Finally, we evaluate the created model's validity.

A. HDD outline

Here, we describe performance verification of the cache function of HDD. The performance of HDD is influenced by the frequency of drive access. For example, while the drive head attainment time (seek time + wait time of revolution) is 16.53msec in the drive of 7200rpm, cache memory control processing needs μ sec order. This proves that time of drive access is dominant in the I/O time of HDD [10]. From this, HDD is equipped with the cache function to hold the accessed data in a memory in order to reduce the number of disk accesses. The utilization efficiency of the cache is improved, and the whole performance is demonstrated.

In this study, we explore and verify the performance of this cache function using model checking and show the results.

Moreover, for the processing time of a drive portion, a value is returned using the time it takes on the average to make data size uniform.

1) Composition of HDD and cache memory

The composition of HDD is shown Figure 6. HDD consists of software, represented by firmware (FW), and hardware, represented by the I/F controller, memory, disk drive, and other controllers.

Next, we explain the processing flow using write processing. First, the HDD receives a host command (workload data) from the I/F controller. Second, the I/F controller sends a command to FW. Third, the FW's cache controller module checks whether writable cache area remains. If it does not, the data on cache is written to the disk drive using a memory controller and drive controller, thus opening up writable space on the cache. Fourth, after writing, new command data is written on cache memory by FW.

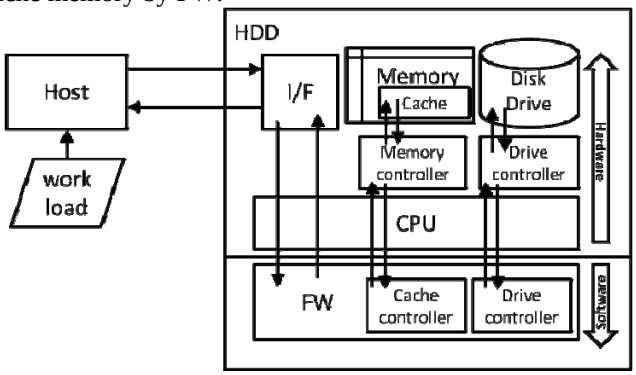


Figure 6. HDD Overview

2) Verification targets

In this paper, we verify the performance of the cache function. Here, performance is defined as execution time.

Based on the above definition, our verified targets define the time from the head command being accepted to the tail command being accepted.

Next, in the future, we plan to use verification results of actual product development. Hence, we plan to make time accuracy of verification results equivalent to the actual system. Therefore, we do not abstract time accuracy.

In this paper, we chose only write processing as the modeling target.

3) Parameters used for cache emulation

Here, we use parameters equivalent to an emulation program. These parameters' information is shown in Table I.

TABLE I. PARAMETERS FOR EVALUATION

Parameter	Meaning
Rotational speed	Revolution per minute
Sector Size	Subdivision area size of a track (512 or 4096 byte)
Cache Size	Total cache size
Average seek time	Head moving time to target
Max segment count	Subdivision area count of Cache memory
Max sector count	Max sector count per track

 ISBN: 978-1-61208-438-1

B. Cache emulation program

Cache processing outlines shown in Figure 7. Before Step 1, the cache program is checked to see if a command has arrived. If it has, cache program is checked to see if it still has easy-to-output data (Step 1). If it does, the cache program transfers the data from cache to a disk drive and opens up writable space in cache (Step 2). If it does not, cache receives a command from the I/F controller (Step 3). Next, the cache program judges whether the new caches used are to be bigger than cache capacity or not (Step 4). If cache overflows, the data chosen by the cache program using a policy (ex: LRU) is written to the disk drive (Step 5). After that, the cache program transfers the data held by I/F to cache memory (Step 6).

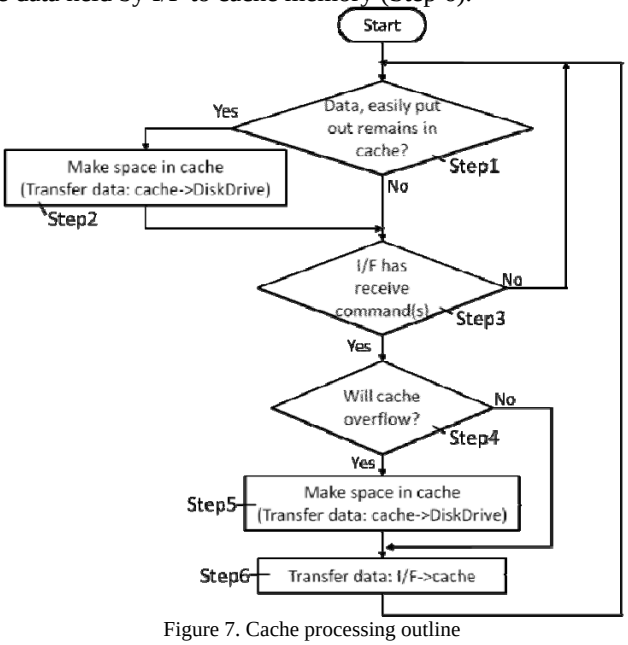


Figure 7. Cache processing outline

On the basis of the above process and in accordance with the modeling plan shown in Section 4-A, we created a verification model written in PROMELA from cache emulation program. Figure 8 shows the state transition diagram of cache emulation program with the object of performance modeling. The emulation program modeling this time does not have a host portion. The module of Host I/F reads the workload file and carries out emulation of cache.

Moreover, to calculate drive access time, we did not use an actual HDD. We use the virtual model that calculates average drive access time in this report.

States of the state transition diagram are as follows. The correspondence state in Figure 7 is shown inside of [].

q0: Workload check [before step1]
q1: Segment count check [step1]
q2: Create drive access list using cache data [step1]
q3: Judge existing access list [step1]
q4: Calculate drive access time and clear cache [step2]
q5: Check exist any drive access [step2]
q6: Set lapsed time by drive access [step2]
q7: Set interval time [before step3]
*q*8: Update system time [before step3]
q9: Obtain commands within update time [step3]
q10: Create new segment [step4]
q11: Modify hit segment [step4]
q12: Check cache size [step4]
q13: Decide destage segment [step5]
q14: Calculate drive access time and clear cache [step5]
q*15*: Transfer data from I/F to cache [step6]
q*16*: Finish

Next, we explain the flow of processing using Figure 8. When workload processing starts, the processing changes to q0: Workload check state. Then, the number of remaining commands of the workload is checked. If there are any remaining commands, the processing will change to q1, and if not, it will change to q16, finish emulation, and verify execution time. In q1: Segment count check state, segment count (Seg) in the cache is checked and whether to output cache contents to the drive or not is determined. If Seg > 1 (outputting cache contents to drive), processing changes to q2. If Seg <=1 (not outputting), then processing changes to state q5. In q2: Create drive access list using cache data state, a drive access list is created and processing changes to q3. In q3: Judge existing access list state, if an access list exists, processing changes to q4. If no list exists, processing changes to q5. In q4: Calculate drive access time and clear cache state, drive access time is calculated and acquired from head LBA address of the access list and the length of access data. After this step is completed, processing changes to q5. In q5: Check if any drive access state exists, check whether existing drive access (at q4 or q14) exists or not. If drive access exists, then processing changes to q6. If not, processing changes to q7. In q6: Set lapsed time by drive access state, drive access time is added to system lapsed time, and processing changes to q8. In q7: Set interval time state, configured interval time is added to system lapsed time, and processing changes to q8.

In q8: Update system time state, system time is updated using set lapsed time. After system time is updated, processing changes to q9. In q9: Obtain commands within update time state, the commands arrive within the updated time. If there are no commands, processing changes to q0. If commands exist, a cache is judged to be a hit or miss. If a command is judged to be a miss, processing changes to q10. If a command is judged to be a hit, processing changes to q11. In q10: Create new segment state, the new segment set up information is secured and processing changes to q12. In q11: Modify hit segment state, the updated information on hit cache segment is acquired and processing changes to q12. In q12: Check cache size state, updated cache size is judged to be bigger than the system cache or not. If it is bigger, processing changes to q13. If not, processing changes to q9. In q13: Decide destage segment state, the segment that is outputted to a disk drive or deleted is chosen by using a scheduling algorithm (ex. LRU), and processing changes to q14. In q14: Cache drive access time and clear cache state, cache segment information and clear segment are outputted and processing changes to q15. In q15: Transfer data from I/F to cache state, the command data which has reached I/F is transfer to cache. After this step is completed, processing changes to q 12.

 ISBN: 978-1-61208-438-1

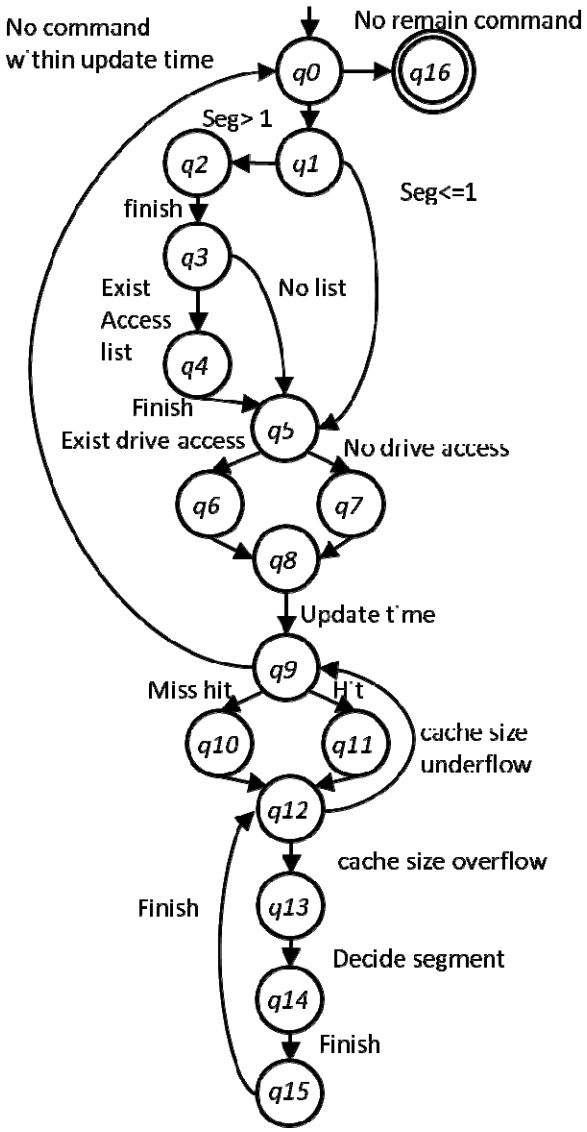


Figure 8. Cache program state transition diagram

The above is processing sequence of the target cache emulation program.

C. Analysis results of cache emulation program

This section describes the analysis result of a cache emulation program. This time, cache performance verification model is created reusing the existing cache emulation C program. Therefore, we describe how to judge whether to reuse the C program part or the new modeling part.

1) Analysis of the cache emulation program based on the contents of verification

Based on the verification contents described in Section 4-A-2, we analyzed the target cache emulation program. This subsection describes the analysis of results.

As described in Section 4-A the HDD I/O performance has dominant disk access time. Additionally, cache processing time does not influence system execution time. Thus, in this verification, addition of lapsed time was limited to the drive access part. However, the opportunity to generate drive access depends on command arrival time. Therefore, we decided to calculate lapsed time on the basis of the command arrival time. Moreover, as mentioned above, since a branch was required to judge the existence of drive processing and a branch accompanying command processing affected lapsed time, they were newly modeled by PROMELA.

Next, from the above-mentioned plan, in processing that determines the contents of drive access, only an execution result influences drive access time, so we thought that the process would not influence performance. Therefore, the processing model that determines the contents of drive access reused the cache emulation C program code. Furthermore, cache emulation program calculates drive access time using only access length, not an internal drive state. Thus, we chose the processing drive portion reusing cache emulation C program code.

From the results of the above analysis, we decided to determine the part that reuses cache emulation C program code and a new modeling part using PROMELA.

D. Development of performance verification model using cache emulation program

1) Create performance verification model

As opposed to the state transition diagram in Figure 8, on the basis of the analysis results in Section 4-C, we decided the part that reuses cache emulation C program code, the part that models using PROMELA, and the part that calculates time progress. The result is shown in Figure 9.

The parts enclosed in a dotted line reuse the existing code, and the parts enclosed in a solid line newly create a model using PROMELA. Time progress processing (to carry out drive access part) is in gray.

The example of modeling in Figure 9 already appeared in Figure 5. Figure 5 shows the same processing as the state diagram that consists of a tri-state of q5, q6, and q7. Lines 1, 2, 6, and 11 in Figure 5 show the same processing as q5. Lines 3 to 5 in Figure 5 show the same processing as q7. Lines 7 to 10 in Figure 5 show the same processing as q6. Finally, lines 3 to 5 and lines 7 to 10 are reused by inserting them into c_code. Other processing parts similarly create a model reusing C code or using PROMELA.

E. The validity check of created model

In this section, the verification model created in Section 4-D is verified using actual work load data. Results are described below.

1) Workload used for verification

In this verification, we use the workload in Table II.

TABLE II. WORKLOAD SPECIFICATIONS

Name	Value
Command count	6510
Command input time range (μ sec)	0~ 35529817
Start LBA range	95~1953512383
Data length (sector)	1~256

2) Parameters for verification

In this verification, we use following parameters shown in Table III.

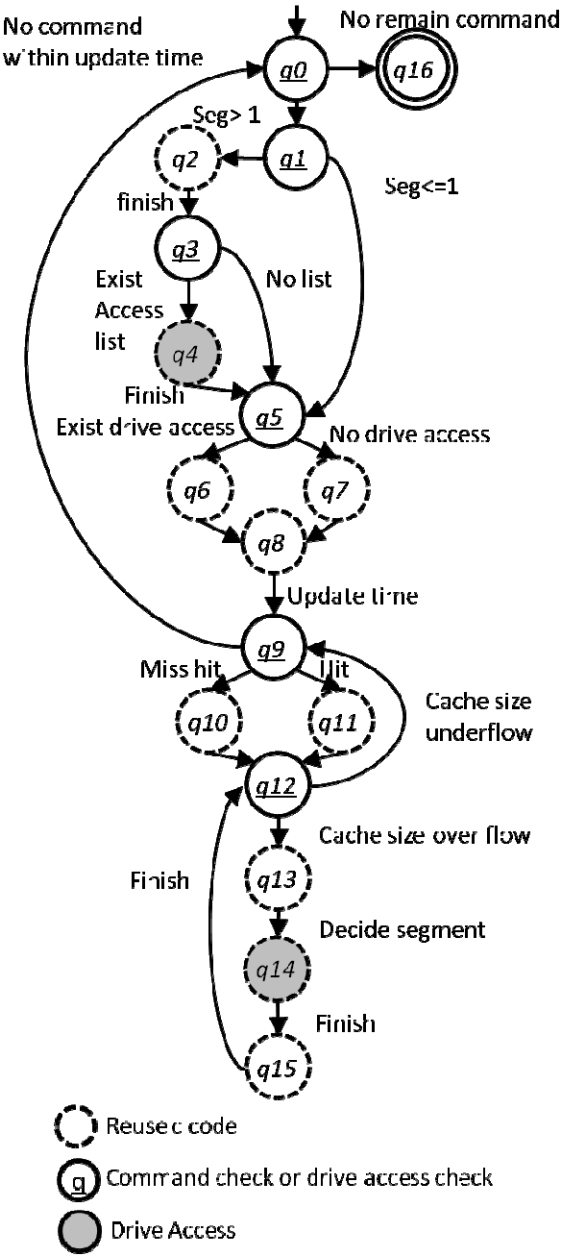


Figure 9. Modeling method

TABLE III. HARDDISK PARAMETERS

Parameter	Meaning
Rotational speed	7200 rpm
Sector Size	512 byte
Cache Size	4,8,16,32,64 MB
Average seek time	8.2 msec
Max segment count	2048
Max sector count	2048

3) PC used for verification

In this verification, we use the PC in Table IV.

TABLE IV. SPECIFICATIONS OF EXPERIMENT PC

Name	Dell Precision T1500
CPU	Intel(R)Core(TM)i7-860 2.8GHz
Memory	16GB DDR3 SDRAM(1066MHz)
Chip Set	Intel(R) H57

4) Using verification tool

In this verification, we use SPIN. The version of used verification tool is SPIN 5.2.5.

F. Verification of execution time

First, we explain the verification of execution time. After the input of the workload, the verification machine calculated execution time and verified whether it satisfied the conditional expression. Then, we verified whether the SystemTime for reaching q16: finish state in Figure 8 exceeded the requirement value. The used verification condition is assert (System Time < Target Time).

A [](System Time < Target Time) can also be used for the same verification.

In the results of this verification, the trail file was outputted when SystemTime exceeded the TargetTime. Thereby, the execution time was verified to satisfy the target or not.

Figure 10 shows an example case in which the above verification conditions were not satisfied.

When the cache size was 4MB and target time was 40,000,000 μ sec, processing took 47,681,370 μ seconds and System Time exceeded requirement time, so a trail file was outputted (Figure 10).

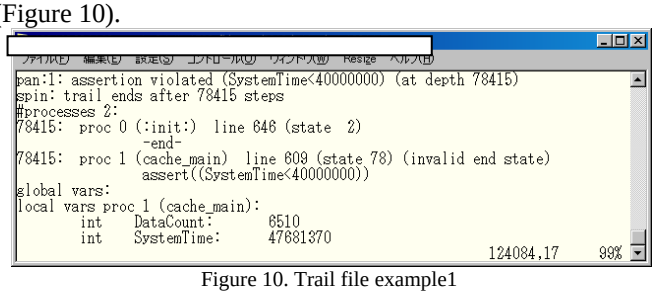


Figure 10. Trail file example1

We acquired the execution results of the cache emulation program and compared them with the verification results of the created model.

The execution results of emulation program are shown in Figure 11.

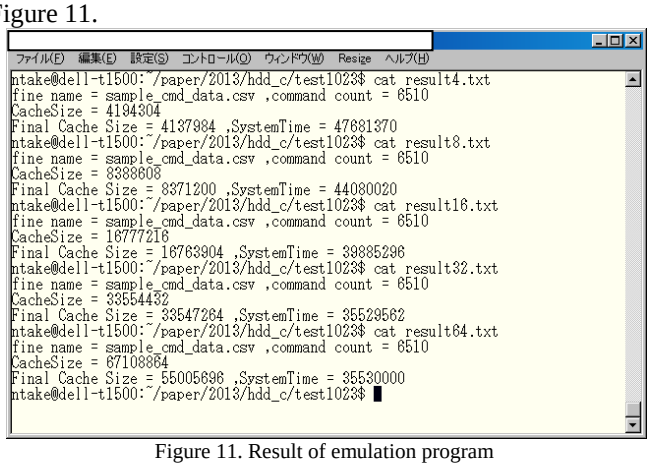


Figure 11. Result of emulation program

The file named result*.txt in Figure 11 is an execution result of an emulation program. The applicable numerical value at * shows the cache size. The result of Figure 10 and the result in cache size equals 4MB of Figure 11 are equivalent. All the results in Figure 11 became equal when a model is executed using the same conditions. From this, the created model was judged to have behavior equivalent to that of an emulation program from this result. As mentioned above, in this research,

 ISBN: 978-1-61208-438-1

the created model was judged to be executed the same as an emulation program. Therefore, the created model is thought to be appropriate.

V. DISCUSSION

A. Source code reuse ratio and evaluation

In this paper, we attempted to create a model more efficient than the newly made model by reusing C source code. Then, we analyzed the ratio of the reused number of C codes close to the number of codes of the model.
The results of analysis are shown in Table V.

TABLE V. RESULTS OF CODE REUSE ANALYSIS

Name	Value
Model LOC	627 (comment lines are excluded)
Cache C code LOC	605 (comment lines are excluded)
C Line in model	363 (Number of C codes (reuse codes) in a model
Reuse rate	57.89% (vs. Model LOC)
Reuse rate	60.00% (vs. Cache C code)

In the results, 60% of original source codes were reused. Moreover, the reuse ratio of the cache C code to a model became 57.89%.

B. Performance exploring using model checking

Next, we show the results of performance exploring using model checking. We used the same verification conditions as described in Ⅳ-F and the code shown in Figure 5, which distributes the cache sizes of 4, 8, 16, 32, and 64MB.

The target time was 40,000,000 μ sec like in Section 4-F, and we carried out performance exploring. In addition, this exploring was completed just to run the program once the pan file that the SPIN generated was executed. Creation of a program as shown in Figure 1 is unnecessary.

The results are shown in Figure 12. These results show that two cache sizes cannot fulfill the conditions, abnormalities occur, and a trail file is generated.

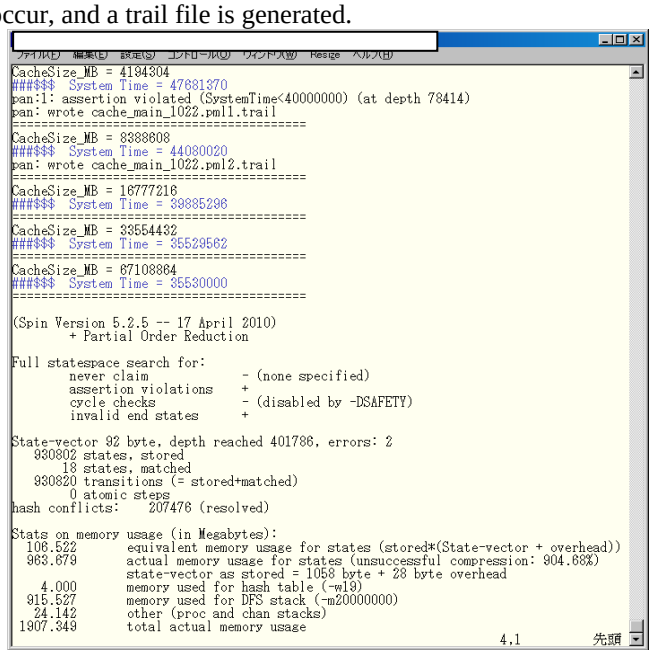


Figure 12. Results of performance exploring

The first pan file has the same contents as Figure 10, so an explanation is omitted. The results of having read the second pan file are shown in Figure 13. As Figure 12 shows, when cache size was 8MB, execution time became 44,080,020 μ sec, which did not satisfy verification formula. In the verification and results in Figure 11, when cache size was less than 8MB, verification showed that target performance could not be attained. It also turned out that 16MB attains target performance with the smallest cache capacity.

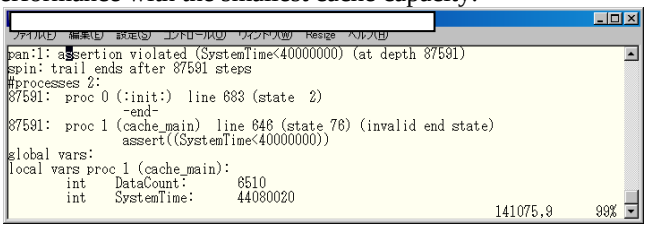


Figure 13. Trail example 2

As mentioned above, in model checking, parameters are explored by using the code for parameter deployment, the code for selection of an algorithm is similarly embedded, and a user becomes able to optimize performance easily.

VI. CONCLUSION AND FUTURE WORK

In this paper, to enable performance exploring for embedded computer systems, which acquire more advanced features and become more complicated every year, we decided to achieve the following objectives for model checking.

- Establish a method for applying model checking to performance exploring
- Develop an efficient performance modeling method

To meet the above objectives, we proposed the following two methods.

1) Easy performance exploring using parameter deployment code

2) Performance verification modeling reusing product code

Moreover, the proposed techniques were applied to a HDD cache emulation program, and we verified whether processing could be completed within a target time and confirmed its validity.

Furthermore, we embedded parameter deployment code to create a performance verification model and achieved performance exploring, and then we the determined that minimum cache capacity required processing was completed within the target time. We also showed that 57.89% of cache emulation program codes were reused to create the new performance verification model. From these results, we validated the proposed technique.
For future work, we need to evaluate whether the proposed technique reduces the man hours in an actual product development.

Moreover, although reuse of code was considered to improve the efficiency of modeling this time, the used part of code will be processed atomically. From the characteristic of HDD, since the criterion of judgment of atomizing was created, it is necessary to also examine the criterion of judgment in the case of applying the proposed technique to other products.

 ISBN: 978-1-61208-438-1

Finally, the performance was defined as execution time and verified in this paper. However, since the throughput is similarly important as an index of performance, it will need to be considered too.

REFERENCES

[1] M. Woodside, G. Franks, and C. Petriu, "The Future of Software Perfo-mance Engineering" in Proc. Future of Software Engineering 2007, May. 2007, pp. 171-187.

[2] C. Smith, L. Williams, "Performance solutions" Addison-Wesley Publishers, 2001.

[3] K. Trivedi, "Probability and Statistics with Reliability" Queuing, and Computer Science Applications. Wiley, 2001.

[4] L. H. Henry, "Software performance and scalability" Wiley, 2009.

[5] Q. Qinru, M. Pedram, "Dynamic power management based on continuous-time Markov decision processes" in Proc. of Design Automation Conference, New Orleans, LA, June 21-25. 1999, pp.555-561.

[6] K. Havelund, A. Skou, K. G. Larsen, and K. Lund, "Formal Modelling and Analysis ofan Audio/Video Protocol: An Industrial Case Study using UPPAAL" in Proc. the 18th IEEE Real-Time System Symposium, Dec 1997, pp 2-13.

[7] T. Nagaoka, A. Ito, K. Okano, and S. Kusumoto, "QoS Analysis of Real-time Distributed System Based on Hybrid Analysis of ProbabilisticModel Checking" IEICE Transactions on Information and Systems, Vol.E94-D, No.5, pp.958-966, May 2011.

[8] K. Moonzoo, K. Yunho, "Automated Analysis of Industrial Embedded Software" in Proc. 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011, pp. 51-59.

[9] R. Alur, D. Dill, "A theory of timed automata," Theoretical Computer Science 126:183-235, April 1994, doi:10.1016/0304-3975(94)90010-8

[10] B. Jacob, N. W. Spencer, D. T. Wang, "Memory Systems Cache, DRAM, Disk" Morgan Kaufmann Publishers,2008

[11] Oracle(R), "Database Performance Tuning Guide 10g Release2" http://docs.oracle.com/cd/B19306_01/server.102/b14211/toc.htm. [Accessed: Sep 24, 2015]

[12] G. J. Holzmann, M. H. Smith, "Software model checking: extracting verification models from source code Formal Methods for Protocol Engineering and Distributed Systems" in Proc. (FORTE/PSTV99) October 1999,pp.481-47.

[13] Compuware, "Applied Performance Management Survey", Oct 2006.

[14] V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization" IEEE Transactions on VLSI Systems, Vol2, pp. 437-445, Dec. 1994, doi:10.1109/92.335012

[15] S. Barber, "Creating Effective Load Models for Performance Testing with Incomplete Empirical Data," in Proc. 6th IEEE Int. Workshop on Web Site Evolution, 2004, PP. 51-59.

[16] A. David, K. Larsen, K. Legay, M. Mikucionis, D. Poulsen, and S. Sedwards, "Runtime Verification of Biological Systems," ISOLA, LNCS, Springer, Vol7609, 2012, pp 388-404.

[17] G. Igna, V. Kannan, Y. Yang, T. Basten, M. Geilen, F. Vaandrager, M. Voorhoeve, S Smet, and L. Somers, "Formal Modeling and Scheduling of Datapaths of Digital Document Printers." Proceedings FORMATS'08, Saint-Malo, France, September 15-17, 2008. LNCS 5215, pp. 170-187.

[18] R. Hamadi, and B. Boualem, "A Petri net-based model for web service composition," Proceedings of the 14th Australasian database conference-Vol17. Australian Computer Society, Inc., 2003, pp 191-200.

[19] Object Management Group, "UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems," http://www.omg.org/spec/MARTE/, [Accessed: Sep 24, 2015]

[20] G. J. Holzmann, "The model checker SPIN ," Software Engineering, IEEE Transactions, Vol23(5), 279-295., May 1997, doi: 10.1109/32.588521

 ISBN: 978-1-61208-438-1

Towards a Better Understanding of Static Code Attributes for Defect Prediction

Muhammed Maruf Öztürk and Ahmet Zengin

Department of
Computer Engineering
Faculty of Computer
and Information Sciences
Sakarya, Turkey 54187
Email: muhammedozturk@sakarya.edu.tr, azengin@sakarya.edu.tr

***Abstract*—Defect prediction requires intensive effort and includes operations which are focused on reducing the cost of software development. These operations involving the use of machine learning algorithms could produce wrong results originated from skewed or missing data. In order to increase the success rate of predictors, defect data sets are either pruned or duplicated. To address this problem, we observe the effects of the derivation of low level metrics using statistical methods in prediction performance. The performance of predictions are evaluated using 10-fold cross-validation on each data set. Experimental results obtained by using 15 data sets show that naive Bayes classifier improved values of Area Under the Curve (AUC) with the rate of 0,1 in average.**

Keywords–Defect prediction; Low level metrics; Metric derivation

I. INTRODUCTION

Properties of software codes vary depending on development processes, functional goals, and development constraints [1][2]. In order to comprehend this variety in depth, we should examine software behaviours and tendencies, in which versions of software changes, along with specific software metric models [3][4][5]. Developers need metric tables to advance their understanding of how software changes across it's newer versions [6]. The standards, which were developed by McCabe and Halstead, are widely used ones while generating software metric tables [7][8]. These standards do not require an in-depth analysis in the structure of codes; however, the model presented by McCabe is more suitable than the others in the design level [9].

Metric tables of software components have a property that indicates the defect-proneness of software. Thanks to this property, a defect prediction can be conducted on the basis of binary classification. However, each data set has potential problems caused by noise or repeated data points that this issue reduces the success of prediction [10]. One of the mostly known problems in defect prediction is class-imbalanced data sets. In such cases, defects are generally intensified on specific parts of software so that the reliability of the prediction is not as desired. In this respect, it is rather difficult to determine a general bias about the software modules [11]. We have two ways to cope with class-imbalance: undersampling, and oversampling. Although undersampling is an efficient method, it causes the hiding of useful data. Likewise, oversampling may cause an unrealistic increase in the success of learning [12], [13][14].

In this study, we investigate metric derivation methods and its effects on defect prediction. Defect data sets consist of 15 data sets including NASA metrics data program (NASA MDP) and Softlab. The common feature of these data sets is that they were generated using McCabe & Halstead metrics. After adding some metrics to the data sets such as character count (cCount) and class size (cS), the variation recorded on the performance parameters such as accuracy and AUC was observed. Moreover, the relationship between low level and other metrics was strived for the exploration. The results obtained from the experiment show that the proposed method increased the success of prediction on 15 data sets in general.

The rest of the paper is organized as follows. Section 2 provides a background describing the relevant terms and approaches. Related works are mentioned in Section 3 and this section also discusses the distinctive aspect of our work when it is compared to similar works. The proposed approach is in Section 4. The results, we have obtained so far, are explained in Section 5. The novelty and the contribution of the paper are presented in Section 6.

II. BACKGROUND

Two types of learning are used in defect prediction: supervised and unsupervised learning. Supervised learning is the most commonly used technique [15][16]. It includes SMV, ANN, decision trees etc.. Although unsupervised learning does not requie a labelling on training data, supervised learning analyzes the data only labeled. Researchers generally want to see which supervised learning techniques are suitable for defect data sets to be predicted. Learning techniques also called predictors are to predict defect-proneness of modules for the next version of software.

Properties of code are prepared using a particular measuring standard namely metrics [17]. Even though researches published in last five years are focused on process metrics that yielded promising results [18][19], code metrics have some gaps that are worthy to explore [20][21]. One of them is

 ISBN: 978-1-61208-438-1

the reliability of defect data sets. As the defect data sets are generally prepared by combining all related developer's comments, they may have missing or noisy data points. In order to cope with this problem, the data are re-sampled or reduced by using particular preprocessing techniques. SMOTE is one of the widely used sampling strategy for defect prediction [22]. However it is sensible to combine a sample reduction method with an over-sampling technique [23].

III. Related Works

One of the leading fields to explore static code properties is machine learning. Menzies et al.'s work, published in 2007, is a much cited work in this field [24]. This work stressed that the type of the metric set is more important than the selected predictor in the success of precision. The promising result of this work is that Bayes classifier showed better performance than J48 with the rate of 71%. Likewise, we have taken naive Bayes among performance measurement algorithms.

The framework developed by Song et al. showed that every data set may not be suitable for every prediction model [25]. This especially changes depending on the type of the data set. Using this result we can say that every learning method is not suitable for every defect data set. A two-phase prediction model was developed in Kim and Kim's work [26], the reports considered as eligible were eliminated in the first phase and the prediction accuracy was obtained as 70%. This work also proved the importance of preprocessing in defect data sets.

One of the works which used NASA MDP data sets is Gray et al.'s work [27]. This work, especially focused on data cleansing, removed some properties of the metrics obtained from 13 data sets to be suitable for binary classification. Missing values were assigned to zero. The first of these results is that used data sets should be extended. Thus, we can determine whether the repeated data points are in general. Second, low level metrics should be used to detect repeated data. Third is the presence of the issues caused by the repeated data.

The studies above all use static code metrics to build a proper prediction model. However, the most relevant work to ours is Gray et al's work which is explained in the preceding paragraph. This work and our work have similarities: they use the same experimental data sets and have claimed the importance of the use of low level metrics.

IV. Proposed Approach

NASA MDP and SOFTLAB data sets consisting of metric values that range from 21 to 40. Tests including ANOVA, t-test, and chi-square unveiled the relationship between character count and LOC (number of lines of code) as below:

$$cCount \sim= lCode * 30. \quad (1)$$

Lorenz and Kidd presented object-oriented metric tables [28]. The main reason why object-oriented metrics are widely used is that such metrics are the best indicator of system reliability at design level [29]. cS is also a low level metric but it is not available in the data sets of NASA MDP and SOFTLAB (CS=total number of operations+ number the attributes) [28]. In order to explore the relationships of defects, Linear and Multiple Regression analyses were used. If the binary-dependencies of the metrics are desired to be extracted, Linear Regression is a convenient method. This method assumes that relations between variables can explained through a linear model [30][31]. Also our approach is to unveil the linear relationships between defect data set values. Given a dependent variable as y-f(x), the assumption having independent variable(x) emerges as y=ax+b. This is called as Curve Fitting [32]. The aim of this process is to find the most suitable a and b variables for f(x). As the value of R^2 closes to the one, a rather suitable curve is obtained. If e_i is regarded as error term, the formula is $e_i = y_{i,measured} - y_{i,model}$. We aim at minimizing S_r in the formula of $S_r = \sum_i^n (e_i)^2$. Linear and nonlinear distribution samples are seen in Figure 1 and Figure 2. The more function curve fits the real data, including large samples up to the count of 17186, the more accurate model is obtained.

If f(x) linear function is to be expressed with more than one independent variable, Multiple Linear Regression is used. For two variables, we have:

$$f(x) = b + a_0 x_1 + a_1 x_2 \quad (2)$$

Our approach can be summarized as follows: 1. The extraction

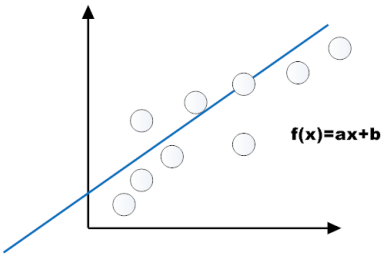


Figure 1. Curve Fitting (Linear).

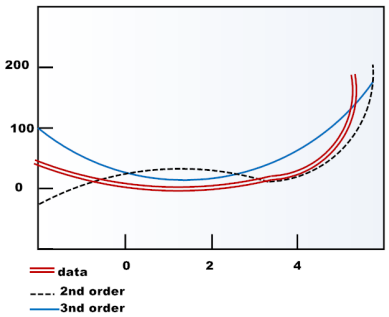


Figure 2. Nonlinear distribution.

of characteristic properties of software defect data sets and exploring required models. 2. The derivation of new low level metrics regarding defect data sets and adding to the data sets. 3. The comparison of data sets including low level metrics with preceding situation.

V. Results

cCount and cS are obtained by using the relationships of data. To test the use of low level metric, we have used 15 data sets including NASA MDP and SOFTLAB. These data sets belong to software projects developed using C, C++, and Java programming languages. The data sets have some metrics range from 21 to 40 including large samples up to the count

 ISBN: 978-1-61208-438-1

of 17186. Data sets, having skewed samples at a certain ratio, comprise 25 missing values. The experimental study has been tested by using the framework we have been developing. This framework is able to generate over the given codes and drives defect prediction with defect prediction algorithms.

The regression analysis results between class size and the other three metrics are illustrated in Figure 3. According to these results, a formula $y = 0,5244x - 14,679, R^2 = 0,9453$ has been found using cS-comment_loc. R^2 is close to one that verifies the consistency of the equation. When it comes to the relation of CS-Executable_loc, an equation is obtained as $y = 9,5518ln(x) - 34,278, R^2 = 0,523$. On the other hand, the effects of Code_and_comment_loc and unique_operand are close to the zero.

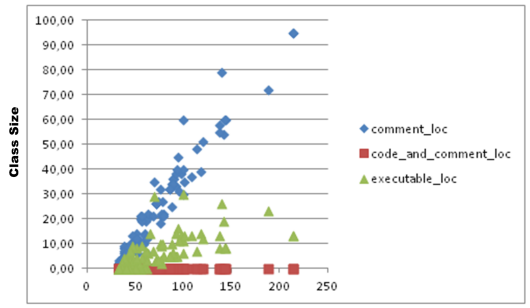


Figure 3. Relations between Class Size and other metrics.

Before the prediction, definitions including defect-prone or not-defect-prone property of software modules should be prepared. If a module does not include any defect and rightly biased then it is labeled as TN. In such cases if the module is wrongly biased then it is labeled as FP. If any module including defects is wrongly biased, labeled as FN. Last, if the bias and the prediction is the same for a defect-prone module, it is labeled as TP. Using these parameters, a table confusion matrix is organized as in Table 1. The success of the proposed method is compared to the others by benefiting the formulas defined in Listing 3.

TABLE I. CONFUSION MATRIX

		PREDICTED	
		nfp	fp
REAL	nfp	TN	FP
	fp	FN	TP

$$Precision = TP/(TP + FP), Recall = TP/(TP + FN) \quad (3)$$

$$TPR = (TP/TP+FN)*100\%, FPR = (FP/FP+TN)*100\% \quad (4)$$

$$Accuracy = TP + TN/(TP + FP + FN + TN) \quad (5)$$

Four classifiers including naive Bayes, Bayes, Random Forest and J48 have been used for the experiment. 10 fold cross-validation has been used along with 10 iteration. One of the evaluation parameters is AUC that is the indicator of the probability of false alarm versus the probability of detection.

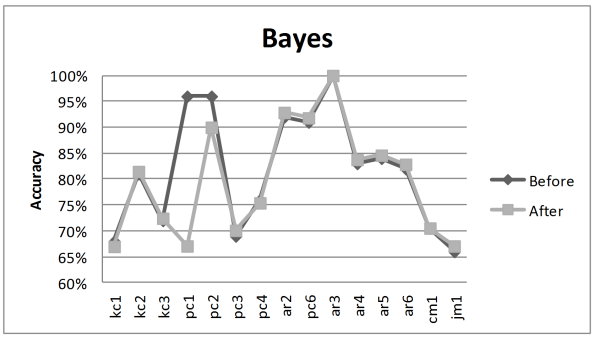


Figure 4. Accuracy values of Bayes.

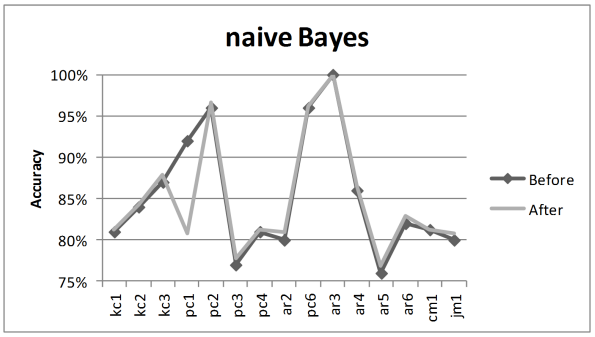


Figure 5. Accuracy values of naiveBayes.

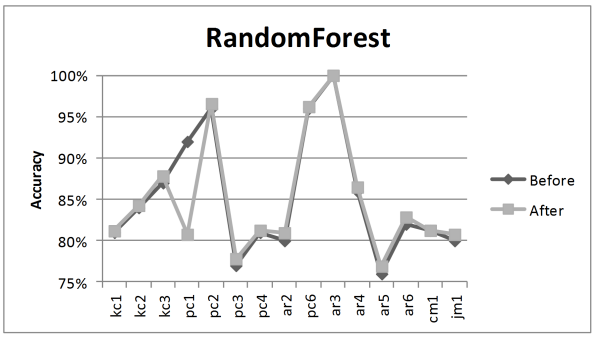


Figure 6. Accuracy values of RandomForest.

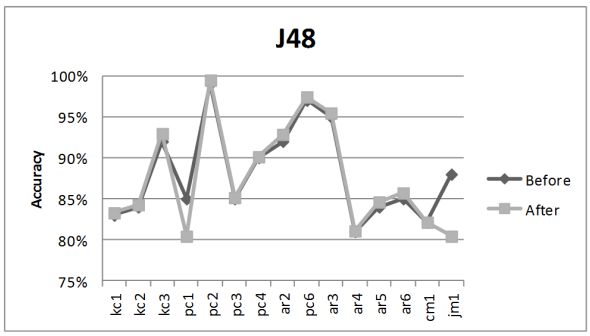


Figure 7. Accuracy values of J48.

 ISBN: 978-1-61208-438-1

On 15 data sets naive Bayes increased the AUC values in general with the rate of 0.1. Figure 4-Figure 7 show some results that explain the successes of the predictors both before the use of low level metrics and after. First, naive Bayes and RandomForest have increased the success of the prediction in all data sets except for the pc1. Second, Bayes has produced worse results than the other algorithms. Last, while the success of J48 on jm1 data set has been reduced, successes of the other algorithms have been increased. Figure 8 and 9 show the AUC values that measures testing reliability. Having low level metrics, remarkable improvement has been achieved on testing set as seen in Figure 9.

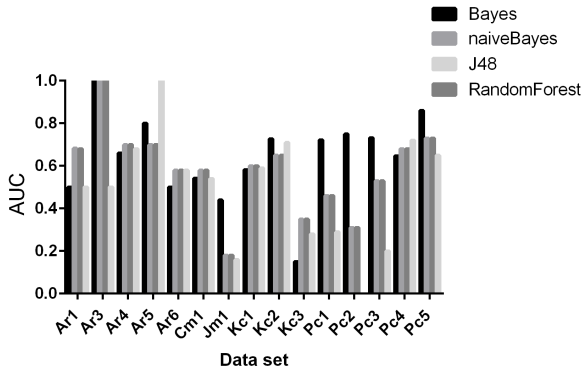


Figure 8. AUC values before preprocessing.

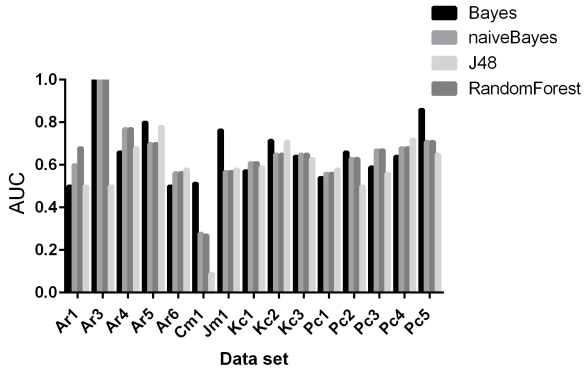


Figure 9. AUC values after preprocessing.

VI. Conclusion

Here, we want to discuss the use of low level metrics in defect prediction and present our approach based on least-square using metric relationships. Thus, extracting mathematical models of the metrics has raised some bias. The first results showed that the use of low level metrics has achieved an unprecedented success in NASA MDP and SOFTLAB data sets.

Low level metrics help us to better understand the details of software systems. However, the success of learning algorithms may not be improved with increasing count of the metrics at steady state. Furthermore, skewness of data sets should be fixed by exposing all data to a preprocessing. To gain better insight, we should develop a preprocessing algorithm which uses some tests such as ANOVA, t-test, and chi-square. In addition, the software, in which data sets are extracted, are coded by using various languages including C, C++, and Java. Therefore, the types of coding should be considered during the extension of metric tables.

The contributions of this paper can be summarized as follows: (i) proposed method for deriving low level metrics could shed new light to researchers in terms of valuable data sets that are not publicly available. (ii) metric relations change depending on the type of coding as in the range of ar3-pc1 coded with C programming language. (iii) using few samples does not produce consistent results such as ar3 data set having 64 samples.

Our current approach has been merely tried on NASA MDP and SOFTLAB data sets. Therefore, one of the purposes which will extend this study is the testing of the approach on other publicly available data sets. An important issue that could arise during the experiment is the ambiguous effects of repeated data points. In this respect, our future work aims to investigate the contribution of the low level metric in the detection of repeated data.

Acknowledgment

The authors would like to thank Tim Menzies who is one of the co-founders of tera-Promise.

References

[1] I. Herraiz, D. Rodriguez, and R. Harrison, “On the statistical distribution of object-oriented system properties,” in Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on. IEEE, 2012, pp. 56–62.

[2] J. Highsmith, Adaptive software development: a collaborative approach to managing complex systems. Addison-Wesley, 2013.

[3] N. Fenton and J. Bieman, Software metrics: a rigorous and practical approach. CRC Press, 2014.

[4] R. J. Leach, Software Reuse: Methods, Models, Costs. AfterMath, 2012.

[5] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, “Choosing software metrics for defect prediction: an investigation on feature selection techniques,” Software: Practice and Experience, vol. 41, no. 5, 2011, pp. 579–606.

[6] L. Putnam and W. Myers, Five core metrics: the intelligence behind successful software management. Addison-Wesley, 2013.

[7] T. J. McCabe, “A complexity measure,” Software Engineering, IEEE Transactions on, no. 4, 1976, pp. 308–320.

[8] M. Halstead, “Potential impacts of software science on software life cycle management,” Purdue University Library, 1977.

[9] T. J. McCabe and C. W. Butler, “Design complexity measurement and testing,” Communications of the ACM, vol. 32, no. 12, 1989, pp. 1415–1425.

[10] L. Pelayo and S. Dick, “Applying novel resampling strategies to software defect prediction,” in Fuzzy Information Processing Society, 2007. NAFIPS’07. Annual Meeting of the North American. IEEE, 2007, pp. 69–72.

[11] G. M. Weiss, “Mining with rarity: a unifying framework,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, 2004, pp. 7–19.

[12] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory undersampling for class-imbalance learning,” Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 2, 2009, pp. 539–550.

[13] T. M. Khoshgoftaar and K. Gao, “Feature selection with imbalanced data for software defect prediction,” in Machine Learning and Applications, 2009. ICMLA’09. International Conference on. IEEE, 2009, pp. 235–240.

 ISBN: 978-1-61208-438-1

[14] T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction." in ICTAI (1), 2010, pp. 137–144.

[15] H. Lu, B. Cukic, and M. Culp, "A semi-supervised approach to software defect prediction," in Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual. IEEE, 2014, pp. 416–425.

[16] H. Lu, E. Kocaguneli, and B. Cukic, "Defect prediction between software versions with active learning and dimensionality reduction," in Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on. IEEE, 2014, pp. 312–322.

[17] C. Kaner et al., "Software engineering metrics: What do they measure and how do we know?" in In METRICS 2004. IEEE CS. Citeseer, 2004.

[18] F. Rahman and P. Devanbu, "How, and why, process metrics are better," in Proceedings of the 2013 International Conference on Software Engineering, ser. ICSE '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 432–441. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486846

[19] I. S. Wiese, F. R. Côgo, R. Ré, I. Steinmacher, and M. A. Gerosa, "Social metrics included in prediction models on software engineering: A mapping study," in Proceedings of the 10th International Conference on Predictive Models in Software Engineering, ser. PROMISE '14. New York, NY, USA: ACM, 2014, pp. 72–81. [Online]. Available: http://doi.acm.org/10.1145/2639490.2639505

[20] P. Oliveira, M. T. Valente, and F. Paim Lima, "Extracting relative thresholds for source code metrics," in Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on. IEEE, 2014, pp. 254–263.

[21] F. Zhang, A. Mockus, I. Keivanloo, and Y. Zou, "Towards building a universal defect prediction model," in Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014, pp. 182–191.

[22] R. Pears, J. Finlay, and A. M. Connor, "Synthetic minority oversampling technique (smote) for predicting software build outcomes," arXiv preprint arXiv:1407.2330, 2014.

[23] L. Chen, B. Fang, Z. Shang, and Y. Tang, "Negative samples reduction in cross-company software defects prediction," Information and Software Technology, vol. 62, 2015, pp. 67–77.

[24] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," Software Engineering, IEEE Transactions on, vol. 33, no. 1, 2007, pp. 2–13.

[25] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," Software Engineering, IEEE Transactions on, vol. 37, no. 3, 2011, pp. 356–370.

[26] D. Kim, Y. Tao, S. Kim, and A. Zeller, "Where should we fix this bug? a two-phase recommendation model," Software Engineering, IEEE Transactions on, vol. 39, no. 11, 2013, pp. 1597–1610.

[27] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Reflections on the nasa mdp data sets," Software, IET, vol. 6, no. 6, 2012, pp. 549–558.

[28] M. Lorenz and J. Kidd, Object-oriented software metrics: a practical guide. Prentice-Hall, Inc., 1994.

[29] Y. Suresh, J. Pati, and S. K. Rath, "Effectiveness of software metrics for object-oriented system," Procedia Technology, vol. 6, 2012, pp. 420–427.

[30] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," Software Engineering, IEEE Transactions on, vol. 38, no. 6, 2012, pp. 1403–1416.

[31] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," in Proceedings of the 5th international Conference on Predictor Models in Software Engineering. ACM, 2009, p. 4.

[32] R. A. Johnson, I. Miller, and J. E. Freund, Probability and statistics for engineers. Prentice-Hall, 2011.

 ISBN: 978-1-61208-438-1

Communication and Coordination Challenges Mitigation in Offshore Software Development Outsourcing Relationships: Findings from Systematic Literature Review

Rafiq Ahmad Khan, Siffat Ullah Khan
Software Engineering Research Group (SERG_UOM),
Department of Computer Science & IT, University of Malakand, Pakistan
E-mail: rafiqahamdk@gmail.com, siffatullah@uom.edu.pk

Mahmood Niazi
Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Saudi Arabia
Faculty of Computing, Riphah International University, Islamabad, Pakistan
E-mail: mkniazi@kfupm.edu.sa

Abstract— **Over the last decade, many firms in the world have started adopting Global Software Development (GSD) in order to reduce software development cost, and access to qualified resources and modern technology. Due to the rapid development of ICTs, the GSD has become an acceptable business strategy with several paradigms. One of the rising business paradigms of GSD is Offshore Software Development Outsourcing (OSDO). The objective of this research is to provide mitigation advice for addressing communication and coordination challenges from vendors' perspectives in OSDO relationships. We have performed systematic literature review (SLR) process for identifying the practices/solutions for these challenges. We have identified 65 practices for addressing these challenges. This paper can help the OSDO vendor organizations to use the identified practices in order to address the communication and coordination challenges in OSDO relationships.**

Keywords—Global Software Development; Software Outsourcing; Communication and Coordination challenges and its Solutions/Practices; **SLR**

I. INTRODUCTION

Many software development companies from the last decade have been trying to enhance their business profits by improving the time-to-market of their products, reducing costs by hiring people from countries with cheaper work-hours. These days, a large number of software development projects are distributed at many different sites and normally located in different countries. This distributed setting of managing a software project is termed as Global Software Development (GSD) and the discipline is termed as Global Software Engineering (GSE) [1]. One of the rising business paradigms of global software development is Offshore Software Development Outsourcing (OSDO) [2]. OSDO represents the practices of holding an outside party to carry out software development work/processes in a state/country other than the one where the products or services are actually developed [3]. Today many software organizations have turned to software outsourcing to get economic cost advantages [4]. Over the last decade outsourcing functions gain competitive advantages due to different reasons, such as the drastic growth in the ICTs market and shortage of information system professionals [4]. In addition, China and India have made the OSDO a reality due to the presence of qualified persons, the availability of resources, skills and better business and economic environment [4].

However, several researchers [5]-[6] recommended that increased globalization of software development creates challenges due to cultural differences, time zone differences, lack of trust, language differences, geographical distance and diversity of communication and coordination. Ali-Babar et al. [7] suggested that the main stumbling block to OSDO is the geographical dispersion. The two major pillars and the backbone of successful OSDO activities are the communication and coordination processes, but it can be hampering due to geographical dispersion, cultural and language differences [8]. The lack of face-to-face meetings is one of the challenges and it affects the process of OSDO [9].

In OSDO relationship, Khan et al. [10] identified various critical challenges faced by vendor organizations. In these challenges, communication and coordination is a critical challenge to vendors in OSDO. Our prior research identified a list of 18 communication and coordination challenges faced by vendors in OSDO relationships [5]. Amongst the identified list of challenges 6 were marked as critical challenges. These identified critical challenges are: 'Geographical Dispersion', 'Cultural Differences', 'Language Differences', 'Lack of Credence', 'Lack of ICT/Technological Cohesion' and 'Lack of Informal/ Face-to-Face Communication' [5].

It is also important to provide mitigation advice in the form of practices for the identified critical challenges as this will help organizations facing these challenges. For this reason, we conducted a SLR process for finding the practices for addressing the aforementioned critical communication and coordination challenges in OSDO outsourcing relationships form vendor's perspectives.

We have formulated the following research question in order to understand the practices/solutions for communication and coordination challenges in OSDO relationships.

RQ. What are the solutions/practices, as identified in the literature, for addressing communication and coordination challenges in OSDO relationships from vendors' perspective?

 ISBN: 978-1-61208-438-1

The structure of the paper is organized as follows: Section II explains the background. Section III explains the research methodology. Results are presented in Section IV. Study limitations are discussed in Section V. Conclusion and future works are presented in Section VI.

II. BACKGROUND

In software outsourcing paradigm, various challenges and hurdles are faced by vendor organizations. Different researchers and practitioners have conducted case studies, questionnaire surveys, focus group sessions, interviews and literature reviews to dig out various aspects of the OSDO relationship.

Alberto Avritzer et al. [11] conducted a case study and suggested that geographic dispersion in global software engineering can be reduced by organizing face to face meetings, effective time management among the team members and "hands-on and Shake-off session", providing possibilities of synchronous communication, giving support for video conference at all sites and also giving suitable selection of communication tools. Cultural differences in OSDO can be reduced by providing the facilities of face to face meeting, cultural training, adopt low-context communication style, cultural liaison/Ambassador and reduce interaction between team from different cultures [12]. The problems of cultural differences can also be mitigated by adapting agile and scrum methods [13]. Similarly the temporal distance in offshore outsourcing can be reduced by establishing a bridging team, relocate to adjacent time zone, adopt and follow the sun development, using appropriate and advance technology, such as ICT, audio and video conferencing, instant messaging, online chat, email, web came and mobile alerts [11].

We can reduce the lack of trust in global software development by managing efficient outsourcing relationships, establishment of an appropriate communication and infrastructure, to encourage effective communication through the adaptation of tools and techniques and promotion of informal communication [12]. The probable solutions of language differences in global software development are composed of translating policies and practices into local languages and by putting emphasis on spoken language skills [14].

The lack of ICT or technological cohesion in global software development can be reduced by using proper communication technologies or tools, such as, internet, video conferencing, data conferencing, teleconferencing, telephone calls, chats, emails, instant messaging, shared databases, Wikis, shared desk top technology, net meeting, change management system, virtual whiteboards, photo gallery, team Intranet websites, electronic meeting systems, voicemail, CAMEL, NEXTMOVE, TAMRI, Dropbox, Mendeley, IRC and Skype etc [15]. Lack of face-to-face or informal communication problems in OSDO relationship can be reduced by provision of multiple communication mode counting support to face-to-face synchronous communication, creation of communication protocols, to promote informal interactions, to apply agile practices (SCRUM), to deploy knowledge transfer mechanisms [16].

By using SLR for identifying the practices/solutions for communication and coordination challenges in OSDO relationships from a vendor's perspective will confine the missing communication and coordination practices in OSDO relationship. The novelty of our research shows that nobody has conducted SLR in this domain to find out practices for addressing communication and coordination challenges faced to vendors in OSDO relationships. The findings will assist OSDO vendor organizations to adopt the identified practices in order to avoid/mitigate the communication and coordination challenges in OSDO relationships.

III. RESEARCH METHODOLOGY

A SLR [17] process was used for data collection, because it is more thorough, less biased, rigorous and open as compared to ordinary literature review [17]. In finding, evaluating and summarizing all available evidences on a specific research question, a systematic review may provide a greater level of validity in its findings than ordinary literature review. A number of researchers [5][18] have used the SLR approach for reviewing the literature. Protocol development is the first phase of the SLR process and it describes planning of the review. In this connection, a systematic review protocol was written first to describe the plan for the review. Details of the various steps in our SLR methodology are available in our SLR protocol [18].

A. Search the Literature

Based on the available access, the digital libraries IEEE Explore, ScienceDirect, ACM Digital Library, SpringerLink and CiteSeer were used to carry out the search phase of the SLR. We used the following search string as a trial search:

((Solutions OR practices OR "best practice" OR "lessons learned" OR Advice) AND ("communication and coordination problems" OR " communication and coordination challenges" OR " communication and coordination norms" OR " communication and coordination barriers" OR " communication and coordination risks") AND ("offshore software outsourcing" OR "information systems outsourcing" OR "IS outsourcing" OR "IT outsourcing" OR "global software development" OR GSD OR "offshore software development outsourcing" OR OSDO))

The major search string was developed and validated after thoroughly getting information and guidance from the trial search. Some digital libraries required different concrete syntax for the search term; we developed the

 ISBN: 978-1-61208-438-1

search string for each resource. In our study, we identify the paper based on the publication's type, such as conference proceeding, databases, specific journals, technical magazines, book chapters, technical books, web pages and reports, etc. In Table I, we represent the final list of resources to be searched also including their search terms and number of publications found in each resource.

TABLE I DATA SOURCES AND SEARCH STRATEGY FOR PRACTICES/SOLUTIONS

S. NO	Resources	Total Results Found	Primary Selection	Final Selection
1	IEEE	1424	166	39
2	Science Direct	1055	82	7
3	ACM	925	114	2
4	Springer Link	347	80	10
5	Cite Seer	500	29	4
Total		4251	471	62

We have selected these resources based on our previous SLRs [5][20] experiences and discussions with our colleagues at the University.

B. Literature Selection

In this section, we are going to presents the criteria for inclusion and exclusion of relevant articles.

a. *Inclusion criteria*

We use the following inclusion criteria for the selection of relevant papers:

- The paper must be relevant to Computer Sciences or Engineering research background because quality research topics in software applications are keep growing from time to time.
- Priority usually goes to journal and conference published papers- that is why in our final selection the majority of papers are journal and conference papers.
- The papers should at least contain challenges, practices and solutions related to communication and coordination in OSDO relationships.
- The papers should contain communication or coordination practices/solutions affecting the continuation or termination of outsourcing relationships.
- Studies that is relevant to outsourcing.

b. *Exclusion criteria*

We use the following exclusion criteria to exclude the irrelevant papers:

- The papers not relevant to Computer Sciences or Engineering research background.
- The studies not relevant to the research questions.
- The papers that are not written in English.
- Studies not mentioned the challenges/ practices/ solutions of communication or coordination in OSDO relationships.
- Studies that contain duplicate data.
- Studies not relevant to outsourcing.

C. Publication Quality Assessment

The publication quality assessment is performed after final selection of publications. During the selection process of studies, some questions were asked to ensure the quality of selected studies. The questions in Table II were constructed to facilitate the studies selection process and to ensure that only relevant papers are being selected. The questions used in the study selection process are shown in the Table II.

TABLE II STUDY SELECTION PROCESS

Question	Answer
Is it clear how communication or coordination practices/solution was measured in OSDO relationship?	Yes/No/Partially
Is it clear how the practices in the selection of software outsourcing vendors were identified?	Yes/No/Partially

By using publication quality assessment questions, studies that are not scholarly reviewed were excluded. Only those studies are selected that aim practices at addressing communication and coordination challenges in OSDO relationships. Similarly, studies that did not provide persuasive results in practices for addressing communication and coordination challenges in the aspects of OSDO relationships were excluded.

D. Data Extraction and Synthesis

The following data was extracted from each publication: Date of review, Title, Authors, Reference, Database, Practices/Solutions: factors that have a positive impact on software development outsourcing vendors, Methodology (interview, case study, ordinary literature review, systematic literature review, report, survey, etc), Target Population, Sample Population, Publication Quality Description, Organization Type (software house, university, research institute etc), Company size (small, medium, large), Country/location of the Analysis and Year.

The data synthesis phase was done by the primary reviewer (the primary author) with the help of secondary reviewer (the co-author). After a thorough review with external reviewer, we have identified 65 practices/solutions from the sample of 62 papers for addressing communication and coordination challenges.

 ISBN: 978-1-61208-438-1

E. Classification of Communication and Coordination practices/solutions

After identifying practices/solutions for addressing communication and coordination challenging in OSDO relationships through SLR, we classified a few practices/solutions in different tables as shown in Section IV. The classification of practices/solutions was based upon the relevant practices/solutions for the identified critical challenges in our previous research [5]. The following criterion for the selection of critical challenges was used:

- Those challenges were considered as critical challenges whose frequency was equal to 40% or higher than 40%. The identified critical communication and coordination challenges are 'Geographical dispersion', 'cultural differences', 'language differences' 'lack of technological cohesion', 'Lack of Informal/Face-to-to face Communication' and 'Lack of Credence'.

IV. RESULTS

This section presents the results of the SLR process for finding the practices/solutions for addressing communication and coordination challenges faced by OSDO vendors.

We identified 65 mitigation advices/practices/solutions for addressing communication and coordination challenges faced to OSDO vendors. SLR has been conducted in the area of OSDO relationships for the identification of these practices. The OSDO vendor organizations can also get help from these practices in order to know that how they can solve the problems of their clients. We have followed SLR guidelines [17] for synthesizing the different practices for the identified critical communication and coordination challenges.

The subsequent sections present the 6 critical challenges and their respective identified practices.

A. Geographical Dispersion

Ali-Babar et al. [7] suggested that the main stumbling block to OSDO is the geographical dispersion. Table III presents the list of our identified 15 practices for addressing the communication and coordination challenge 'Geographical Dispersion'.

TABLE III PRACTICES FOR ADDRESSING GEOGRAPHICAL DISPERSION

CCCC1: Geographical Dispersion		
S/NO	***Practices/Solutions for Addressing Geographical Dispersion***	***% of Practices via SLR (N=62)***
1	Use of technology to make knowledge sharing easier between the teams. Such as, webcams and instant messaging software to improve communication and coordination between the team members distributed across multiple sites	50
2	Synchronous communication, such as face-to-face meetings, online chats, teleconferences, and web conferences, is ideal for quick status meetings, brainstorming sessions, and reviews. Asynchronous communication, such as email, discussion forums, and shared documents, provides a persistent record of discussions and decisions, and don't require participants to be available at the same time	47
3	Shifting the working hours of both the onshore and offshore teams, by adjusting direct meetings to the time zones or by creating asynchronous meetings via project managers.	23
4	Communicate with clients timely	23
5	Negotiate teams working hours for Synchronicity	21
6	Create a team calendar aiding in project planning	18
7	Encourage both asynchronous and synchronous communication	15
8	Establish communication guidelines, technical infrastructure for information and communication, for example, effective tools and work environments	15
9	Provides opportunities for synchronous interactions without prior schedule definition	15
10	Be online or stay connected	6
11	Assign technical lead to each site that would be responsible to coordinate process, development and schedule activities	3
12	Create bridging team	2
13	Create roles, relationships and rules to facilitate coordination and control over geographical, temporal and cultural distance	2
14	Promote visits and exchanges among sites	2
15	Utilize the global distribution to conduct tasks ''over night'', e.g. the test of new components so that the results are available on the following morning	2

B. Cultural Differences

Cultural differences is a critical challenge faced in the communication and coordination processes because it can slow down the OSDO activities [20].

TABLE IV PRACTICES FOR ADDRESSING CULTURAL DIFFERENCES

CCCC2: Cultural Differences		
S/NO	***Practices/Solutions for Addressing Cultural Differences***	***% of Practices via SLR (N=62)***
1	Establish open communication between stakeholders through face to face meetings, instant messaging and onsite visits	57
2	Use of online tools for online team-building if visits won't work	49
3	Arrange training and workshops to understand both client organization and people culture involved in OSDO	31
4	Define a cultural ambassador for the project to create teams with complementary skills and cultures	13
5	Create close cooperation between team members involved at both client and vendor side to built trust-worthy relationship	8
6	Build mixed teams with memberships from different cultural backgrounds.	7
7	Create roles, relationships and rules to facilitate coordination and control over geographical, temporal and cultural distance	7
8	Increase project members' domain knowledge and	5

	reduced cultural distance by using Agile Methods	
9	Introduce a neutral third-party Agile coach	5
10	Appoint strong leadership for each team	5
11	Make visible the work progress for all stakeholders	4
12	knowledge of the client's language and culture	4
13	Take equality and justice approach in management activities.	2

Table IV presents the list of our identified 13 practices for addressing the communication and coordination challenge 'Cultural Differences'.

C. Lack of Credence

Several researchers [5][12][20] recommended that increased globalization of software development creates challenges due to cultural differences, time zone differences, lack of trust, language differences, geographical distance and diversity of communication and coordination.

TABLE V PRACTICES FOR ADDRESSING LACK OF CREDENCE

CCCC3: Lack of Credence		
S/NO	***Practices/Solutions for Addressing Lack of Credence***	***% of Practices via SLR (N=62)***
1	Investing in building and maintaining trust and good relations	30
2	Arrange frequent meetings in various forms such as video conferencing, personnel rotations, and team building exercises	21
3	Improve vendor's capability such as technical, managerial, and staffing capabilities as this play a cardinal role in maintaining a client's trust in an ongoing business relationship.	18
4	Improve personal relationship with clients	15
5	Promote efficient outsourcing relationship	13
6	Promote informal meetings	10
7	Effective and frequent communication between clients and vendors at all levels of the organizational hierarchy are pivotal for managing trust	10
8	Build efficient a contract and Conform to the contract and quality of deliverables	9
9	Spending resources on reducing socio-cultural distance by means of facilitating face-to-face meetings.	9
10	Implement the contract successfully is it was signed without cost overrun etc.	5
11	Have at least some people at each node who have met people at peer nodes in person. This also reduces the perceived geographical distance, if not the physical. This helps promote trust and reduce fear	4
12	Early and frequent delivery of working software	4
13	Travel to client location for establishing friendly ties	4
14	Use status (every three weeks) to signal transparency	4
15	Run series of workshops	2
16	Using Scrum practices in GSD improved communication, trust, motivation and product	2
17	Use Trusty, a tool which was designed to support the distributed software development process	2

Table V presents the list of our identified 17 practices for addressing the communication and coordination challenge 'Lack of Credence'.

D. Language Differences

The two major pillars and the back of OSDO relationships are the communication and coordination processes, but it is not properly achieved due to several challenges like geographical dispersion, culture, time zone and language differences [8].

TABLE VI PRACTICES FOR ADDRESSING LANGUAGE DIFFERENCES

CCCC4: Language Differences		
S/NO	***Practices/Solutions for Addressing Language Differences***	***% of Practices via SLR (N=62)***
1	Use of communication media to support a sense of co-located and synchronous interaction by employing facial expressions, body language, and speech	50
2	Understand the language and business culture of clients	12
3	Encourage face-to-face meetings	10
4	Select a vendor with knowledge of the client's language	7
5	Review project document by a native speaker	4
6	Encourage team members to use standard language/common language in order to avoid miss-interpretation	2
7	Appoint team members having fluency in English language	2
8	Appoint language translator	2

Table VI presents the list of our identified 8 practices for addressing the communication and coordination challenge 'Language differences'.

E. Lack of Informal/Face-to-face Communication

Lack of face to face meetings is raised due to the parties being widely dispersed from each other, and hence it affect the process of OSDO [9]. Table VII presents the list of our identified 14 practices for addressing the communication and coordination challenge 'Lack of Informal/Face-to-face Communication'.

TABLE VII PRACTICES FOR ADDRESSING LACK OF INFORMAL/FACE-TO-FACE COMMUNICATION

CCCC5: Lack of Informal/Face-to-Face Communication		
S/NO	***Practices/Solutions for Addressing Lack of Informal/Face-to-Face Communication***	***% of Practices via SLR (N=62)***
1	Adopt appropriate communication tools like videoconferencing, Teleconferencing, Data Conferencing and Web-Based Technologies	52
2	Encourage frequent communication through latest technologies	50
3	Daily exchange of the project status by technologies such as, telephone calls, video conferences or emails etc	50
4	Create a Communication Protocol	15
5	Increase frequency of communication between team	15

 ISBN: 978-1-61208-438-1

	members	
6	Create team having technical skills and cultural awareness	10
7	Establish cooperation by to one member from each team. This might possibly solve some of the communication decencies, e.g., when decisions are made at informal meetings.	9
8	Arrange conferences/workshops for distributed team members	7
9	Build trustworthy relationship	7
10	Sponsor team members for site visits	4
11	Create a database that contains the areas of expertise of the individual project participants	4
12	Arrange weekly conference calls by the central team or the remote team(s) to talk about the status of the project and clarify questions, or they take place at dates specified in the project plan, usually to discuss deliverables	2
13	Use Distributed Agile models e.g. SCRUM	2
14	Use of tools such as 'Trusty' to support software development process	2

F. Lack of ICT/Technological Cohesion

Communication and coordination processes in OSDO relationships can be hampered due to high cost and lack of ICT [12].

TABLE VIII PRACTICES FOR ADDRESSING LACK OF ICT/TECHNOLOGICAL COHESION

CCCC6: Lack of ICT/Technological Cohesion		
S/NO	***Practices/Solutions for Addressing Lack of ICT/Technological Cohesion***	***% of Practices via SLR (N=62)***
1	Adopt Different Latest Technologies such as: Teleconferencing (two-way audio) e.g., NetMeeting, CU-SeeMe, Microsoft Exchange, Dropbox, Wikis, Mendeley etc. Videoconferencing (two-way audio and video) e.g., NetMeeting, CU-SeeMe, Microsoft Exchange, Dropbox, Wikis, Mendeley Data Conferencing (whiteboards, application sharing, data presentations) e.g., NetMeeting, Evoke, WebEx, etc. Web-Based Technologies Tools (Intranets, Listservs, Newsgroups, chat, message boards) e.g., E-groups, Yahoo Groups, Open Topics, etc. Proprietary (with or without web browser interface) e.g., Lotus Notes, IBM Workgroup, ICL Team WARE Office, Novell GroupWise, The Groove, etc. Voice over IP Electronic Meeting Systems e.g., Group Systems, Meeting Works, Team Focus, Vision Quest, Facilitate.com, etc.	52
2	Adopt both Asynchronous (text) and Synchronous (voice) tools like: Telephone, E-mail, Instant Messaging, Wiki, Internet, Voicemail, Shared Databases, Mailing lists, IRC, Messenger, Skype, Chat, Phone, Net meeting, Change Management System, Virtual white boards, Photo Gallery, Team Intranet Websites, Group Calendars, Fax, Power Point Presentations, Blog, Nor-real-time database, CAMEL, NEXT MOVE, TAMARI and Team space	50
3	Arrange ICT Training Sessions for the team members	10
4	Use of Web Technologies for Collaboration e.g. Web-based tutoring, web-based mentoring, web-based knowledge mining and web-based knowledge profiling	5
5	Arrange Knowledge Sharing Activities between team members	5
6	Arrange social events for awareness between team members	5
7	Build Communication Protocol	4
8	Adopt Distributed Agile Models such as Distributed pair programming and Urgent request	4

Table VIII presents the list of our identified 8 practices for addressing the communication and coordination challenge 'Lack of ICT/Technological Cohesion'.

V. STUDY LIMITATIONS

By using the SLR process, we have extracted data about the practices/solutions for addressing communication and coordination challenges; however, we might have omitted some practices? For internal validity, one possible threat is that any specific article may have not in fact described underlying reasons to report practices/solutions for addressing these challenges. This threat has not been independently controlled by us. Other threat is publication bias during SLR process. By using our SLR process, we may have missed out some relevant papers, due to the increasing number of papers in software outsourcing. However, like other researchers of SLR, this is not a systematic omission [21].

How valid are our findings? The results of our finding are not based on studies that used a random sample of software developing outsourcing organization in the world. Yet, in the exploration of our research question, our study is the most comprehensive up to date. As discussed in result sections, the dilemma of simplifying our findings can also be measured by evaluating the finding of other related studies. To provide support for simplification, we found many similarities in our findings as compare to other people's findings. In order to decrease the researcher's bias, we have carried out the inter-rater reliability tests in the selection of primary studies and data extraction phases. Due to limited resources and not enough access to every digital library, we were unable to find out all the relevant papers in our area, although, the used digital libraries are sufficient for the simplification of findings in our study.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have provided mitigation advice in the form of practices for addressing communication and coordination challenges from vendors' perspectives in OSDO relationships. Our results reveal that focusing on these practices can help vendor organizations in order to strengthen their relationships with client organizations in OSDO. However, we recommend independent studies on this topic in global software development. This will increase confidence in our results and also track changes in attitudes to OSDO activities over time. We have identified the

following goals that we plan to follow in future from the findings of this study:

- The practices/solutions for addressing communication and coordination challenges will be validated using empirical studies with practitioners working in outsourcing industries, as done by other researchers [22][23].
- The practices/solutions in OSDO relationships from client's perspectives will be analyzed.

Our future work will focus on developing a Communication Coordination Challenges Mitigation Model (CCCMM). This paper gives only one component of the CCCMM, such as the identification of various practices/solutions for addressing communication and coordination challenges via SLR. The proposed CCCMM will bring together and advance the work that has been undertaken on frameworks and models for outsourcing relationships.

REFERENCES

[1] R. Britto, V. Freitas, E. Mendes, and M. Usman, "Effort Estimation in Global Software Development:A Systematic Literature Review," in IEEE 9th International Conference on Global Software Engineering, Shangai, China, 2014, pp. 135-144.

[2] P. Lago, H. Muccini, and M. Ali-Babar, "Developing a course on designing software in globally distributed teams.", IEEE International Conference on Global Software Engineering, ICGSE, Bangalore , 17-20 Aug, 2008, pp. 249-253

[3] M. Ali-Babar, J. M. Verner, and N. P. Thanh, "Establishing and Maintaining Trust in Software Outsourcing Relationships: An Empirical Investigation," The Journal of Systems and Software, vol. 80, no. 9, 2007, pp. 1438-1449.

[4] D. Avison and T. Gholamreza, "Outsourcing and Offshoring Information System Projects," Information Systems Project Management, p. 351: SAGE Publications, Inc., 2009.

[5] R. A. Khan and S. U. Khan, "Communication and Coordination Challenges in Offshore Software Development Outsourcing Relationship from Vendors' Perspective: Preliminary Results," ISoRIS2014 Malaysia, Special edition, Journal of Science International Lahore, vol. 26, no. 4, 15-16 October, 2014, pp. 1425-1429.

[6] S. Mehmood, M. Niazi, and H. Akthar, "Identifying the Challenges for Managing Component-Based Development in Global Software Development: Preliminary Results," in Proceedings of the Science and Information Conference (SAI 2015), 2015, pp. 933 – 938.

[7] M. Ali-Babar and L. Christian, "Global software engineering: Identifying challenges is important and providing solutions is even better," Information and Software Technology, vol. 56,2014, pp. 1-5.

[8] I. Richardson, "A Process Framework for Global Software Engineering Teams " Information and Software Technology, 2012, vol 45 (11), pp. 1175-1191.

[9] M. Hansen and H. Baggesen, "From CMMI and Isolation to Scrum, Agile, Lean and Collaboration.", Agile Conference, 2009. AGILE '09., Chicago, IL, 24-28 August, 2009, pp. 283-288.

[10] S. U. Khan, M. Niazi, and R. Ahmad, "Critical Barriers for Offshore Software Development Outsourcing Vendors: A Systematic Literature Review " in Software Engineering Conference, APSEC '09, Asia-Pacific 2009, pp. 79 - 86

[11] A. Alberto, B. Sarah, K. Josiane, S. Menasche, N. John, and P. Maria, "Survivability Models for Global Software Engineering," in IEEE 9th International Conference on Global Software Engineering, Shangai, China, 2014, pp. 100-109.

[12] J. Verner, O.P. Brereton, B. A. Kitchenham, M. Turner, and M. Niazi, "Risks and risk mitigation in global software development: A tertiary study," Information and Software Technology, vol. 56, 2014, pp. 54–78.

[13] P. Maria and L. Casper, "Could Global Software Development Benefit from Agile Methods?," International Conference on Global Software Engineering, ICGSE '06, Florianopolis, Oct 2006, pp. 109-13.

[14] S. Wu, "Overview of Communication in Global Software Development Process. IEEE Inernational Conference on Service Operations and Logistics, and Informatics (SOLI), Suzhoe, 8-10 July, 2012. pp. 474-478"

[15] G. Vanessa and M. Sabrina, "Problems? We All Know We Have Them. Do We Have Solutions Too? A Literature Review on Problems and Their Solutions in Global Software Development," in IEEE Seventh International Conference on Global Software Engineering, Porto Alegre, 27-30 Aug. 2012, pp. 154-158.

[16] M. Niazi, "An Instrument for Measuring the Maturity of Requirements Engineering Process," Product Focused Software Process Improvement, vol. 3547, 2005, pp. 574-585.

[17] B. Kitchenham and S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering Keele University and Durham University Joint Report, 2007, pp 1-44.

[18] M. Niazi, "Do Systematic Literature Review Outperform Informal Literature Reviews in the Software Engineering Domain? An Initial Case Study," Arabian Journal for Science and Engineering, vol. 40(3), March 2015, pp. 845-855.

[19] R. A. Khan and S. U. Khan, "Communication and Coordination Challenges in Offshore Software Outsourcing Relationships: A Systematic Literature Review Protocol," Gomal University Journal of Research, vol. 30, no. 1, 2014, pp. 9-17.

[20] S. U. Khan and M. I. Azeem, "Intercultural Challenges in Offshore Software Development Outsourcing Relationships: An Exploratory Study Using a Systematic Literature Review," IET Software, vol. 8, no. 4, 2014, pp. 161-173.

[21] E. Hossain, M. Ali-Babar, and H. Y. Paik, "Using Scrum in Global Software Development: A Systematic Literature Review.," in IEEE International Conference on Global Software Engineering, ICGSE09, Lero, Limerick, Ireland., 2009, pp. 175-184.

[22] M. Niazi, K. Cox, and J. Verner, "An empirical study identifying high perceived value requirements engineering practices," in Advances in Information Systems Development, Fourteenth International Conference on Information Systems Development (ISD´2005) Karlstad University, Sweden, 2006, pp. 731-743.

[23] S. Mehmood, "Empirical Study of Software Component Integration Process Activities," IET Software, vol. 7, no. 2, 2013, pp. 65 – 75.

 ISBN: 978-1-61208-438-1

Adapting Heterogeneous ADLs for Software Architecture Reconstruction Tools

Dung Tien Le

Thai German Graduate School of Engineering,
King Mongkut's University of Technology North Bangkok
Bangkok - Thailand
Email: le.t-sse2013@tggs-bangkok.org

Ana Nicolaescu, Horst Lichter

Software Construction Research Group
RWTH-Aachen University
Aachen - Germany
Email: ana.nicolaescu@swc.rwth-aachen.de,
horst.lichter@swc.rwth-aachen.de

***Abstract*—Architecture reconstruction tools were proposed to enable the extraction of descriptive architecture models based on prescriptive input models. A limitation of these tools is that they employ specific meta-models to which the input prescriptive models must adhere. These are often incompatible with the languages or notations that architects use in practice, leading to substantial effort to overcome terminology differences, to transform possibly already existing prescriptive models in tool-compatible ones and interpreting the results. To alleviate this problem we propose to leverage model engineering techniques in order to enable heterogeneous prescriptive and descriptive models as input and output artifacts of reconstruction tools. We exemplify our proposal by extending the Architecture Analysis and Monitoring Infrastructure (ARAMIS) - an approach developed within our previous work for the reconstruction and evolution of software architectures with a strong focus on the behavior view.**

Keywords–Software Architecture; Architecture Reconstruction; Model-To-Model Transformation; Architecture Description Language; Unified Modeling Language.

I. INTRODUCTION

It is generally acknowledged that the architecture greatly affects the quality of a given software and that its description is crucial to support understanding, decision making, etc. For example, Bass et al. stated that the software architecture is essential because of three main reasons: it is the basis for communication among stakeholders, it encompasses the important, early design decisions and it is a transferable abstraction of a system [1]. Because of its importance, over the years numerous attempts and even standards [2] have been proposed to support the description of the software architecture. A plethora of methods, tools and languages covering a very wide spectrum of formality were proposed and used to serve this purpose. Architects often use informal descriptions in the form of text, boxes and lines diagrams and alike but also employ more formal languages like the Architecture Description Languages (ADLs) or Unified Modeling Language (UML) when more formality is needed and/or required. Nowadays, there are more than 100 published ADLs available for use [3]. The use of UML to describe architectures has also increased, especially after the introduction of UML Profiles in UML 2.0 [4]–[7]. When considering the wide pallet of choices and the uncertainty regarding their suitability for use in a given context, it can seem natural to consider unifying these in one single highly-expressive architectural language. However, in a recent journal publication [8], Malavolta et al. stated that such an universal language is unlikely to become popular. Instead, each architectural language will be created based on specific stakeholders requirements.

Due to the numerous possibilities to describe architectures, their purpose and the various involved stakeholders, it is common that even in the same project or company, the software architecture is described differently using various tools and languages. Typically, most of the effort to document architectures is invested in the early phases of the software development process and the result thereof is the so-called "prescriptive architecture". Although descriptions are in later phases very useful to support the system's further development, these usually go out of date soon because of the relatively high effort that should otherwise be invested to keep them consistent to the actual architecture [9].

To approach this problem, several architecture reconstruction (AR) techniques were proposed. These aim to identify "the descriptive architecture" which is the actual description that reflects the system's reality. In order to use these approaches, usually the architects must specify the prescriptive architecture in advance. The descriptive architecture model is then derived by correcting the prescriptive one with information extracted from the real system. However, for defining the prescriptive architecture, the architects are bound to use the meta-model of the employed AR tools [10]. These meta-models are usually stiff and cannot be extended. For example, even though the architects have initially used UML Profiles or a given ADL to describe a prescriptive architecture, if the tool that they currently want to employ only defines layers, then the architects must re-describe the architecture using only this concept. As our previous work has shown [10], this situation can lead to misunderstandings and in the end, prohibit the wide adoption of the considered AR tool. While the meta-models of other AR tools are extendible, there might still be gaps between what the architects are familiar with and the new meta-models. Furthermore, effort must be invested in order to understand and extend a given AR meta-model.

In our opinion, there is a need for reconstruction tools that address this **heterogeneity problem**. The architects should be able to model the prescriptive architecture using their familiar languages or tools. Then, by employing such an AR tool, a descriptive architecture model should be retrieved that adheres to the same meta-model as the prescriptive one. In this paper, we present an approach to extend the ARAMIS Workbench - developed during our previous work to evaluate the communication between the architecture units composing software systems - with the possibility to allow the input and output of heterogeneous prescriptive and descriptive architecture descriptions respectively.

This paper is structured as follows: in Section II, we present the ARAMIS concepts that are the foundations of

 ISBN: 978-1-61208-438-1

the ARAMIS Workbench. Section III presents our solution to enable different types of architecture descriptions within ARAMIS. Section IV discusses the related work and Section V concludes the paper.

II. ARAMIS

The Architecture Analysis and Monitoring Infrastructure (ARAMIS) is "*a tool-supported framework for run-time monitoring, communication integrity validation, evaluation and visualization of the behavior view of software architectures*" [11], [12]. ARAMIS allows the architects to validate the communication between the hierarchies of architecture units that constitute a given system. In order to do so, ARAMIS maps extracted low-level run-time traces on architecture units and validates the mapped communication according to the rules given in the prescriptive architecture. The ARAMIS meta-model (ARAMIS-MM) [12] to which the prescriptive architecture should adhere to, although developed for flexibility is still specific. The ARAMIS Workbench offers technical mechanisms for the mapping and validation of the communication and the visualization of the result using various interactive views.

One of the major limitations of this concept is that both the prescriptive and descriptive architecture models must adhere to the ARAMIS-MM. The visualizations are also ARAMIS-specific. This leads to situations in which architects must first (1) re-describe their prescriptive, e.g., component-based diagram using the ARAMIS Architecture Modeller and then (2) interpret the result as displayed in an ARAMIS-specific visualization that has no traceability links with their prescriptive architecture model from step (1).

In order to loosen this limitation and increase the acceptability of ARAMIS, we currently work on enhancing ARAMIS so that it allows flexible input and output architecture descriptions. In such a scenario the architect would merely upload, e.g., a component diagram and receive as output the same diagram, augmented with run-time information (e.g., frequency with which one component accesses another one) and information regarding occurred architecture violations.

III. Goals and Solution Concept

Our main goals that we pursue with our approach are:

- enable the architects to reuse their prescriptive architecture models even though these might not necessarily conform to ARAMIS-MM.
- enable the generation of outputs that conform to the same meta-model as the input. Preferably, the output should be obtained by simply augmenting the prescriptive input model, in order to boost understanding by leveraging recognition effects.

In order to solve the heterogeneity problem mentioned in the introduction, we developed a solution concept to fill in the gap between the popular architectural languages - that are being used by the architects - and ARAMIS. The core of the concept is to transform an existing architecture description (AD) of a software system into an AD that conforms to the ARAMIS meta-model and subsequently to reverse the transformation to present the output.

Model-to-model (M2M) transformation is the process of producing one or more output (target) models based on one or more input (source) models. Based on the modeling languages used for the input and output, we can differentiate between two types of transformation: *exogenous* - the input and output languages are different, *endogenous* - the input and output languages are the same [13]. To enable the transformation, a so-called transformation definition consisting of transformation rules must be created. The transformation rules are specified at meta-model level and prescribe how one or more elements from the output model must be produced based on one or more elements from the input model. Upon performing the actual transformation, the application of these rules leads to the emergence of transformation links between the elements of the input and output models. If the transformation rules are bidirectional, then the transformation is also named bidirectional, otherwise it is called unidirectional.

A M2M transformation suitable to solve the problem described before must be (1) exogenous - because the input models are probably not ARAMIS-specific - and (2) unidirectional. Given that the ARAMIS-MM is very general (see Figure 1), we assume that the probability that more

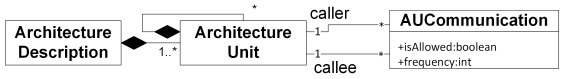


Figure 1. Excerpt from the ARAMIS-MM

elements from the input meta-model (e.g., box, component) must be transformed to the same ARAMIS-MM element (e.g., architecture unit) is relatively high. In such a scenario, defining bidirectional transformation rules can be complex. Instead, in order to enable the architects to analyze the result on their own architecture description, we propose to store the concrete links resulted during the transformation and reuse them after the ARAMIS validation results are available in order to map these on the input model. This leads to the same effect as the bidirectional transformation.

Another important aspect deals with the nature of the input and output models. The output model expresses the architecture from a behavior point of view. More explicitly, the communication of two architecture units is assigned a frequency and is possibly marked as a violation. *If the input model offers a structural overview of the architecture, then there are probably no dedicated meta-model elements to express these behavioral aspects.* There are at least two options to address this problem. We can either *reuse general purpose elements with a loose semantic from the input meta-model* (e.g., in UML we can append the results using UML *comments*) or, alternatively, we can *extend the input meta-model with additional suitable elements*, (e.g., a new property called "frequency" can be added to an already existing element called "DirectedLine").

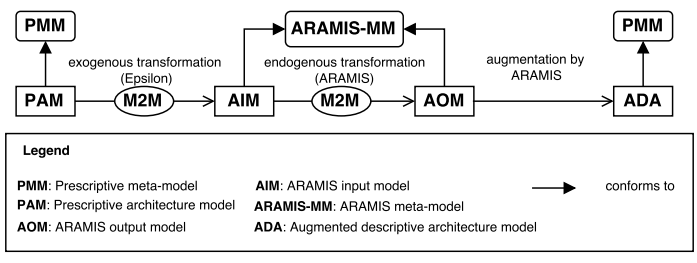


Figure 2. The model transformation chain in ARAMIS.

In order to enable the exogenous, unidirectional transformations, we implemented a solution that uses Epsilon

 ISBN: 978-1-61208-438-1

[14], a fully integrated environment for model engineering that, among other features, supports meta-model design, creating inter-model links, generating model editors, and M2M transformations. Also, because of its active community and provided documentation with comprehensive examples, its learning curve is reduced. The model transformation chain that resulted when we extended ARAMIS with our Epsilon-based solution is represented in Figure 2.

To use Epsilon, we first converted the ARAMIS-MM [11] into an equivalent Ecore model representation. When a prescriptive architecture model with a new, previously not analyzed meta-model must be considered, this meta-model must first also be documented in an Ecore model and then the transformation rules between it and the ARAMIS-MM can be defined. By applying the transformation rules, a set of transformation links emerges.

Assuming that each model element can be uniquely identified and differentiated (e.g., by its ID), we can keep track of every transformation with the exact source and target model elements. The result of the transformation, i.e., the ARAMIS input model, will further undergo a subsequent endogenous transformation performed by the ARAMIS Workbench which will then create the ARAMIS output model. This endogenous transformation creates an important issue: the ARAMIS output model might contain elements that were not present in the input model and thus are not linked to the prescriptive model (e.g., unforeseen communication between architecture units). This issue is a sign of a mismatch between the prescriptive and descriptive architecture. The architects can use this result to further investigate the considered software system.

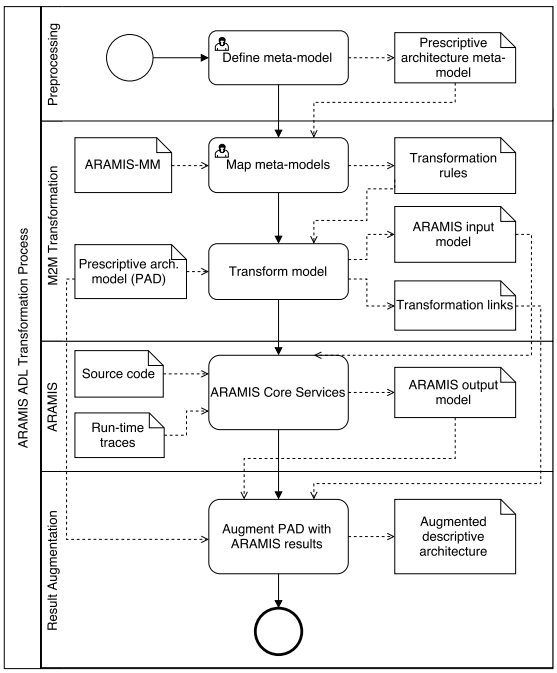


Figure 3. The ARAMIS ADL transformation process

The process encompassing all activities necessary to employ ARAMIS using a new ADL is depicted in Figure 3. This process consists of four major steps: 1. Preprocessing, 2. Model to Model Transformation, 3. ARAMIS Processing, 4. Result augmentation.

In the following, we exemplify the steps 1,2 and 4 using an example based on a simple boxes-and-lines ADL. Step 3 is not further detailed in this paper, since it was covered by our previous work [11].

A. Preprocessing

To exemplify our approach, we have implemented an example using a *boxes-and-lines* ADL. Figure 4 shows its meta-model (BL-MM). As mentioned above, the preprocessing step prepares the prescriptive meta-model (in this case BL-MM) for the next steps, by creating a corresponding Ecore meta-model.

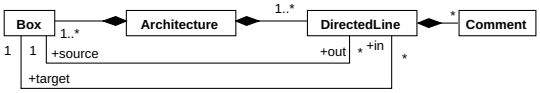


Figure 4. Meta-model of a simple boxes-and-lines ADL

In this case, an extension of the BL-MM that permits the addition of behavior-related information is not necessary, because we can use for this purpose the *Comment* BL-MM element. The validation results from ARAMIS,i.e., the frequency of the calls and their validity, will be augmented in the initial model using *Comment* elements.

B. Model transformation

Based on the ARAMIS-MM (see Figure 1) and the BL-MM, we define the *transformation rules*. In our example, we want to create transformation rules for (1) mapping *Box* in BL-MM on *ArchitectureUnit* in the ARAMIS-MM and (2) mapping *DirectedLine* of BL-MM on *AUCommunication* of ARAMIS-MM. Figure 5 presents a simple boxes-and-lines

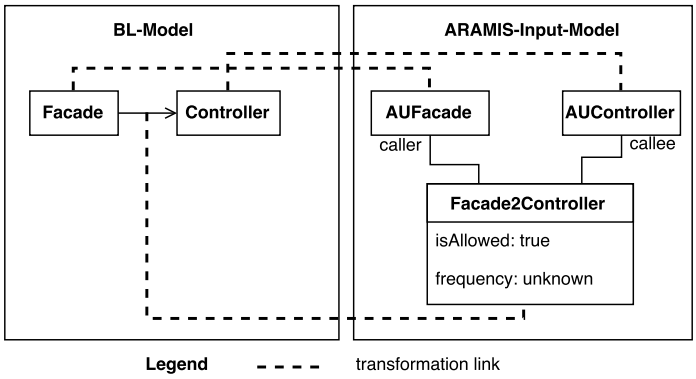


Figure 5. Example of a model transformation

model (on the left hand side) and the ARAMIS model elements that are the result of the M2M transformation. Facade and Controller are transformed to the *AUFacade* and *AUController* respectively. The call from Facade to Controller is transformed to an AUCommunication *Facade2Controller* that has *AUFacade* as caller, *AUController* as callee, an initial frequency *unknown* and a *true* isAllowed attribute. These correspondences are then saved as transformation links.

C. Augmenting the ARAMIS results

The ARAMIS output model is presented on the left side of Figure 6. First, we can see that, after running ARAMIS, *Facade2Controller* now has the updated *frequency* value of *100*. Second, there is a new element that did not exist in the input model: *Controller2Facade*. This element appears because ARAMIS detected that AUController has also accessed the

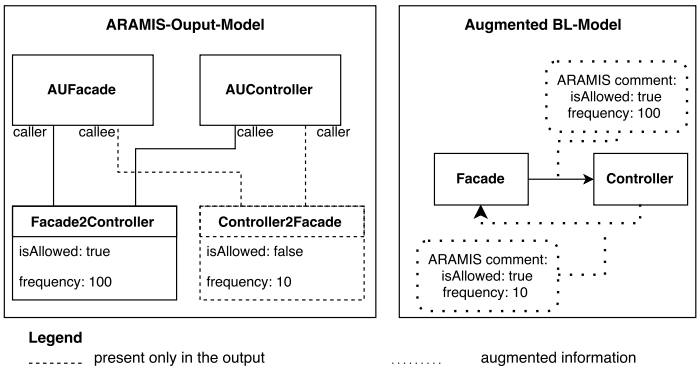


Figure 6. Example: ARAMIS augmented result

AUFacade during run-time. Based on this result, the prescriptive architecture model is augmented. Based on the previously generated transformation links, we know that our M2M transformation transformed the prescriptive model element *Facade* into the ARAMIS *AUFacade*; transformed *Controller* into *AUController*; and transformed the directed line between *Facade* and *Controller* into *Facade2Controller*. We can now use this information to augment the prescriptive model. For this we create in the input model a new Comment element that we attach to the DirectedLine from Facade to Controller as shown in the left side of Figure 6. This comment contains the *isAllowed* and *frequency* attributes that characterize the communication between *Facade* and *Controller*. Furthermore, since there is no transformation link for the ARAMIS Controller2Facade, a new DirectedLine is added to the initial model for the detected communication from Controller to Facade. To this, we also attach a corresponding comment with information regarding its frequency and permission.

IV. Related work

Most of the architecture reconstruction tools have rigid architecture description meta-models. For example, Sonargraph-Architect [15] allows users to define the architecture of the software systems using layers, layer groups, vertical slices, vertical slices groups and subsystems. The architects cannot use other types of ADL.

Malavolta et al. [16] proposed the **DUALLY** framework that supports architectural and tools interoperability. By using its intermediate ADL meta-model for architectural language, it provides ADL interoperability, but no support for architecture reconstruction or validation is available.

The meta-model of the SoftArch reconstruction tool includes 3 architecture concepts: *components*, *associations* and *annotations*. The users can then create customized figures for the various elements, to simulate the use of various meta-models [17].

V. Conclusion and future work

In this paper, we presented an approach for enabling heterogeneous input and output architecture descriptions for the ARAMIS Workbench. We have implemented an extension for ARAMIS to leverage a M2M transformation using the Epsilon framework. Our solution aims to close the gap between the ADLs that the architects are familiar with and ARAMIS. To reduce the amount of time/complexity for further model transformations, we are offering pre-defined transformation rules for the most popular cases, such as boxes-and-lines and UML component diagrams. In the future we plan to evaluate our solution within an extensive case-study on a real-world system.

An open question related to our work is how to reduce even more the effort needed to be invested by the architects when using ARAMIS. For example, if the input boxes and lines diagram is simply a drawing, we currently expect that the architect ”translates” the diagram to an Ecore model. A complete solution would employ image recognition techniques to directly transform the model. Given that different techniques might be necessary depending on the type and form of input model, this represents an important limitation of our approach.

References

[1] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[2] “ISO/IEC/IEEE 42010,” http://www.iso-architecture.org/42010 [accessed: 2015.10.01].

[3] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, “The up-to-date list of currently existing architectural languages,” http://www.di.univaq.it/malavolta/al/ [accessed: 2015.10.01].

[4] J. Pardillo and C. Cachero, “Domain-specific language modelling with UML profiles by decoupling abstract and concrete syntaxes,” Journal of Systems and Software, vol. 83, no. 12, Dec. 2010, pp. 2591–2606.

[5] M. H. Kacem, A. H. Kacem, M. Jmaiel, and K. Drira, “Describing dynamic software architectures using an extended uml model,” in Proceedings of the 2006 ACM Symposium on Applied Computing, ser. SAC ’06. ACM, 2006, pp. 1245–1249.

[6] P. Selonen and J. Xu, “Validating uml models against architectural profiles,” in Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2003, pp. 58–67.

[7] N. Medvidovic, D. S. Rosenblum, D. F. Redmiles, and J. E. Robbins, “Modeling software architectures in the unified modeling language,” ACM TOSEM, vol. 11, no. 1, Jan. 2002, pp. 2–57.

[8] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, “What industry needs from architectural languages: A survey,” IEEE Trans. Softw. Eng., vol. 39, no. 6, Jun. 2013, pp. 869–891.

[9] S. Ducasse and D. Pollet, “Software architecture reconstruction: A process-oriented taxonomy,” IEEE Trans. Softw. Eng., vol. 35, no. 4, Jul. 2009, pp. 573–591.

[10] A. Dragomir, M. F. Harun, and H. Lichter, “On bridging the gap between practice and vision for software architecture reconstruction and evolution: A toolbox perspective,” in Proceedings of the WICSA 2014 Companion Volume. ACM, 2014, pp. 10:1–10:4.

[11] A. Dragomir, H. Lichter, J. Dohmen, and H. Chen, “Run-time monitoring-based evaluation and communication integrity validation of software architectures,” in Proceedings of the 2014 21st Asia-Pacific Software Engineering Conference - Volume 01, 2014, pp. 191–198.

[12] A. Nicolaescu, H. Lichter, A. Göringer, P. Alexander, and D. Le, “The aramis workbench for monitoring, analysis and visualization of architectures based on run-time interactions,” in Proceedings of the 2015 European Conference on Software Architecture Workshops, ser. ECSAW ’15. ACM, 2015, pp. 57:1–57:7.

[13] M. Brambilla, J. Cabot, and M. Wimmer, Model-Driven Software Engineering in Practice, 1st ed. Morgan & Claypool Publishers, 2012.

[14] “Epsilon,” http://www.eclipse.org/epsilon/ [accessed: 2015.10.01].

[15] “Sonargraph Architect,” https://www.hello2morrow.com/products/sonargraph/architect [accessed: 2015.10.01].

[16] I. Malavolta, H. Muccini, P. Pelliccione, and D. Tamburri, “Providing architectural languages and tools interoperability through model transformation technologies,” IEEE Trans. Softw. Eng., vol. 36, no. 1, Jan. 2010, pp. 119–140.

[17] J. Grundy and J. Hosking, “Softarch: Tool support for integrated software architecture development,” International Journal of Software Engineering and Knowledge Engineering, vol. 13, 2003, pp. 125–152.

 ISBN: 978-1-61208-438-1

Verifying and Constructing Abstract TLA Specifications: Application to the Verification of C programs

Amira Methni*, Matthieu Lemerre†, Belgacem Ben Hedia†, Serge Haddad‡ and Kamel Barkaoui*
*CNAM, CEDRIC, 292 rue Saint Martin, Paris Cedex 03, France
Email: first.last@cnam.fr
†CEA, LIST, Centre de Saclay, PC172, 91191, Gif-sur-Yvette, France
Email: matthieu.lemerre@cea.fr, belgacem.ben-hedia@cea.fr
‡LSV, ENS Cachan, CNRS & INRIA, France
Email: haddad@lsv.ens-cachan.fr

***Abstract*—One approach to verify the correctness of a system is to prove that it implements an executable (specification) model whose correctness is more obvious. Here, we define a kind of automata whose state is the product of values of multiple variables that we name State Transition System (STS). We define the semantics of TLA+ (specification language of the Temporal Logic of Actions) constructs using STSs, in particular the notions of TLA+ models, data hiding, and implication between models. We implement these concepts and prove their usefulness by applying them to the verification of C programs against abstract (TLA+ or STS) models and properties.**

Keywords–Temporal Logic of Actions; formal specification; model-checking; C programs; refinement mapping.

I. INTRODUCTION

As software systems become large and error-prone, formal verification methods become an essential key concept to ensure their correctness. Model Checking [1] provides an automated technique to check and detect errors in computer programs. But despite its promise, the verification process may be complex due to the size of these systems. One useful technique to reduce the complexity of verification process is abstraction. Generally, an abstract model specify "what" the system do while the concrete model describes "how". The idea is to map the concrete set of states to a smaller set of states resulting in an approximation of the system with respect to the property of interest. We say that the concrete model implements the abstract one. Verifying the abstract model is generally more efficient than verifying properties of the original.

a) Contributions: We define an operational semantics of a TLA specification in terms of automata, that we called State Transition System (STS). We remind the concepts of implementation relation and refinement mapping in TLA+ that we formalize in terms of relations between STSs. The refinement between specifications can be checked with the TLC model checker. Verified properties on the abstract specification can thus be deduced in the concrete specification. A way to abstract details of the concrete specification is to hide its irrelevant variables. TLA+ can express data hiding, but TLC can't support this type of construct. So, we have implemented the notion of data hiding by constructing a STS that we call "quotient STS", which is constructed by extending the TLC model checker. In order to let the quotient STS be analyzed by existing tools, we extend the TLC model checker to produce an LTS that can be checked by the CADP toolkit. We apply the mentioned concepts on C programs using our tool C2TLA+. Preliminary results show the importance of using an abstract model to reduce the complexity of verification.

b) Outline: The remainder of the paper is structured as follows. We give an overview of TLA+ and its operational semantics in Section 2. Section 3 reminds the concepts of refinement mapping and the implementation relation between specifications and describe a way to construct the quotient STS. In Section 4, we apply these concepts to verify the correctness of the C implementation with respect to their specification and we report some preliminary experimental results obtained. We discuss related work in Section 5. Section 6 concludes and presents future research directions.

II. AN OPERATIONAL SEMANTICS FOR TLA SPECIFICATION

In this section, we explain some basics concerning the syntax and the semantics of TLA [2]. Then, we describe the operational semantics of TLA using a STS.

A. Overview of TLA+

TLA+ is a formal specification language based on the TLA [3] for the description of reactive and distributed systems. TLA itself is a variant of linear-time temporal logic. The semantics of TLA is defined in terms of *states*. A *state* is a mapping from variables to values. A *state function* is a nonboolean expression built from constants, variables and constant operators, that maps each state to a value. For example, $y + 3$ is a state function from a state s to three plus the value that s assigns to the variable y. An *action* is a boolean expression containing constants, variables and primed variables (adorned with "$\prime$" operator). Unprimed variables refer to variable values in the actual state and primed variables refer to their values in the next-state. Thus, an action represents a relation between an old state and a new state. For example, $x = y' + 2$ is an action asserting that the value of x in the old state is two greater that the value of y in the new state. A *state predicate* (or predicate for short) is an action with no primed variables.

Syntactically, TLA formulas are built up from actions and predicates using boolean operators ($\neg$ and $\wedge$ and others that can be derived from these two), quantification over logical variables ($\forall$, $\exists$), the operators $\prime$ and the unary temporal operator $\Box$ (*always*) of linear-time temporal logic [4].

The expression $[A]_{vars}$ where $\mathcal{A}$ is an action and $vars$ the tuple of all system variables, is defined as $\mathcal{A} \vee (vars' =$

 ISBN: 978-1-61208-438-1

$vars$). It states that either $\mathcal{A}$ holds between the current and the next state or the values of $vars$ remain unchanged when passing to the next state. For any action $\mathcal{A}$, the state predicate $Enabled(\mathcal{A})$ describes whether the action $\mathcal{A}$ can be executed in the current state s, i.e., there exists some state t such that $s \longrightarrow t$ is an $\mathcal{A}$ step.

To specify a system in TLA, one describes its allowed behaviors. A *behavior* is an infinite sequence of states that represents a conceivable execution of the system. The system specification can be given by the temporal formula Φ defined as a conjunction of the form:

$$\Phi \triangleq Init \wedge \Box[Next]_{vars} \wedge F \quad (1)$$

Where, $Init$ is the predicate describing all legal initial states, $Next$ is the next-state action defining all possible transitions between states and F is a conjunction of fairness assumptions about the execution of actions. However, other forms of specification are possible and can occasionally be useful.

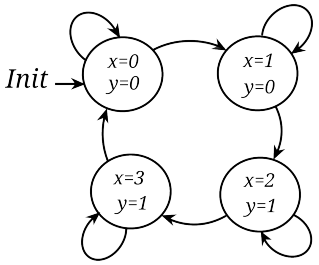
A TLA formula is true or false on a behavior, which is a sequence of states. Let $\sigma = \langle s_0, s_1, \ldots \rangle$ be a behavior. σ satisfies $Spec$ iff $Init$ is true of the first state s_0 and every state that satisfies $Next$ or a "stuttering step" that leaves all variables unchanged.

B. State Transition System

In TLA, the behavior of a system is modeled as an infinite sequence of states. The operational semantics of a TLA specification can be given in terms of a STS, which is easier to work with than sets of sequences.

$$\begin{array}{lll} \Phi & \triangleq & \wedge\ (x = 0 \wedge y = 0) \\ & & \wedge\ \Box\ [\ \ \wedge\ \ x' = (x+1)\%4 \\ & & \qquad\ \wedge\ \ y' = x \div 2 \qquad\quad]_{\langle x,y \rangle} \end{array}$$

(a) TLA specification



(b) The STS of Φ

Figure 1. The operational semantics of a TLA specification

Definition 1: A STS is a 3-tuple $\mathcal{T} = (\mathcal{Q}, \mathcal{I}, \delta)$ given by

- a finite set of states $\mathcal{Q}$,
- a set $\mathcal{I} \subseteq \mathcal{Q}$ of initial states,
- a transition relation $\delta \subseteq \mathcal{Q} \times \mathcal{Q}$.

Figure 1 shows a TLA specification and its corresponding STS $\mathcal{T}_\Phi = (\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \delta_\Phi)$ which encodes all its possible behaviors ($\div$ symbol denotes integer division). The specification Φ is translated into $\mathcal{T}_\Phi$ as follows:

- $\mathcal{T}_\Phi$ has initial state(s) $\mathcal{I}_\Phi$ specified by the predicate $x = 0 \wedge y = 0$,
- every state $s \in \mathcal{Q}_\Phi$ corresponds to a valuation of the state function $\langle x, y \rangle$,
- each transition $t \in \delta_\Phi$ corresponds to satisfying the predicate $[x' = (x+1)\%4\ \wedge y' = x \div 2]_{\langle x,y \rangle}$.

III. Refinement and Abstraction of TLA Specifications

A way to reduce the verification task is to define an abstract model as a specification, and then relate behaviors of the abstract model to those of the implementation. Properties checked on the abstract model can be deduced on the concrete one. We use concrete model to refer to high-level specification and abstract model to refer to low-level specification. This section describes the semantics of refinement between a high-level and a low-level TLA+ specification. Then, we present a way to automatically construct a reduced model, which abstracts the detailed behavior of the concrete TLA+ specification.

A. Refinement Mapping

Abadi and Lamport [5] described that a high-level specification Ψ implements a low-level specification Φ iff for each behavior of Ψ, there is a behavior of Φ with the same sequence of externally visible states, allowing stuttering, e.g., if the state Φ does not change during a finite number of steps. This implementation relation is proved by defining a refinement mapping between specifications.

Let Ψ and Φ be two TLA specifications, $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ the variables occurring in the specifications Ψ and Φ respectively. A (concrete) specification Ψ implements an abstract specification Φ if $\Psi \Rightarrow \Phi$. The proof of this implication consists in defining state functions $\bar{y_1}, \ldots, \bar{y_n}$ in terms of the variables $y_1, \ldots, y_n$ and prove that $\Psi \Rightarrow \bar{\Phi}$, where $\bar{\Phi}$ denotes the formula Φ obtained by substituting $\bar{y_i}$ for the free occurrences of y_i, for all i.

The set of state functions $\bar{y_1}, \ldots, \bar{y_n}$ is called a *refinement mapping*. The "barred variable" $\bar{y_i}$ is the state function with which Ψ implements the variable y_i of Φ. So, if σ is the behavior $s_1 \rightarrow s_2 \rightarrow s_3 \ldots$ of Ψ, we define the behavior $\bar{\sigma}$ to be $\bar{s_1} \rightarrow \bar{s_2} \rightarrow \bar{s_3} \ldots$ We say that Ψ implements Φ under this refinement mapping iff, for each behavior σ satisfying Ψ, the behavior $\bar{\sigma}$ is a behavior of Φ.

B. Implementation Relation and Property Preservation

The proof $\Psi \Rightarrow \Phi$ under a refinement mapping is sufficient to verify that Ψ implements Φ [5]. The key to the implication relation is that TLA allows to write only formula that are *insensitive to stuttering*, i.e., given a TLA formula Φ and two stuttering equivalent runs σ and σ', Φ holds along σ if and only if it holds along σ' [3]. This implementation relation between TLA specifications can be viewed as a weak simulation relation between its corresponding STSs.

Definition 2: Let $\mathcal{T}_\Psi = (\mathcal{Q}_\Psi, \mathcal{I}_\Psi, \delta_\Psi)$ and $\mathcal{T}_\Phi = (\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \delta_\Phi)$ denote two STSs. A simulation $\mathcal{R}$ relation from $\mathcal{Q}_\Psi$ to $\mathcal{Q}_\Phi$ is a function that satisfies the following conditions:

- $\forall s \in \mathcal{I}_\Psi, \mathcal{R}(s) \subseteq \mathcal{I}_\Phi$ (initial states are mapped to initial states),

- For each state pairs $(s_1, s_2) \in \delta_\Psi$, $(\mathcal{R}(s_1), \mathcal{R}(s_2)) \in \delta_\Phi$ (state transitions are mapped into state transitions or stuttering steps).

If a lower-level specification, expressed by a TLA formula Ψ, implements an abstract specification Φ, Ψ preserves all TLA properties of Φ if and only if for every formula ϕ, if $\Phi \Rightarrow \phi$ is valid, then so is $\Psi \Rightarrow \phi$. This is true if $\Psi \Rightarrow \Phi$.

C. Data Hiding in TLA

A very useful form of data abstraction is variable hiding, which refers to providing only essential information to the outside world and hiding not needed information. In TLA, it is possible to hide some variables using the existential quantifier $\exists$ (which differs from the quantifier of predicate logic). The formula $\exists\, x : \Phi$ asserts that it doesn't matter what the actual values of x are, but there are some values x can assume for which Φ holds. The meaning of $\exists$ is defined by (2). The formula $\sigma \sim_x \tau$ is defined to be true iff σ can be obtained from τ (or vice-versa) by adding and/or removing stuttering steps and changing the values of x. Thus, the (2) is true for a behavior σ iff Φ is true for some behavior τ such that $\sigma \sim_x \tau$ is true.

$$\sigma \models \exists\, x : \Phi \triangleq \exists_{\ behavior}(\sigma \sim_x \tau) \wedge (\tau \models \Phi) \qquad (2)$$

The temporal formula (3) describes a specification Φ where v is the list of all relevant state variables and x is the list of internal (hidden) variables.

$$\Phi \triangleq \exists\, x : Init \wedge [Next]_v \wedge L \qquad (3)$$

The existential operator is a very simple and useful way in which the system is described as a black box. However, in practice, the TLC model checker cannot handle the TLA hiding operator. In what follows, we present a way to implement data hiding by constructing a quotient STS from a TLA specification.

D. Computing a Quotient STS

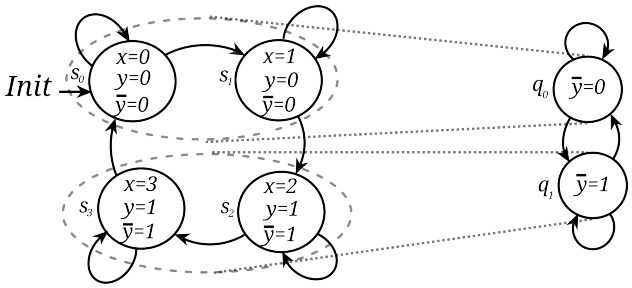


Figure 2. Constructing the quotient STS using the refinement mapping $\bar{y} \triangleq y$

Given a concrete STS $\mathcal{T} = (\mathcal{Q}, \mathcal{I}, \delta)$ describing a TLA specification, one can obtain an abstraction of $\mathcal{T}$, a small STS that we call *quotient STS* and which is obtained by quotienting the states $\mathcal{Q}$ under a refinement mapping γ.

Figure 2 shows a STS resulting from adding a refinement mapping $\bar{y} \triangleq y$ in all states of the concrete STS. The quotient STS (at the right side of the figure) is constructed by collapsing all states related under the relation γ into the same state. Let $\mathcal{T}/_\gamma = (\mathcal{Q}/_\gamma, \mathcal{I}/_\gamma, \delta/_\gamma)$ be the quotient STS of $\mathcal{T} = (\mathcal{Q}, \mathcal{I}, \delta)$ under the refinement mapping γ. The algorithm of constructing $\mathcal{T}/_\gamma$ is given in Figure 3.

```
1:  procedure QUOTIENTSTS
2:      Q_γ ← γ(I)
3:      NotSeen ← {s' ∈ Q | s ∈ Q and (s, s') ∈ δ}
4:      while NotSeen ≠ {} do
5:          for ∀q ∈ NotSeen do
6:              if γ(q) ∉ Q_γ then
7:                  Q_γ ← Q/_γ ∪ {γ(q)}
8:                  δ/_γ = δ/_γ ∪ {(γ(q), γ(q')) | (q, q') ∈ δ}
9:                  NotSeen = NotSeen \ {q}
10:             end if
11:         end for
12:     end while
13: end procedure
```

Figure 3. Construction algorithm of the quotient STS

We extend the implementation of TLC to produce the quotient STS "on-the fly" when the TLC model checker computes the state space of a specification. In fact, TLC makes efficient use of disk. It doesn't keep all states in memory which is the limiting factor of the explicit other model checkers. Instead, it stores just fingerprints of states, which is a 64-bit number generated by a "hashing" function. So, the probability that two states have the same fingerprint is 2^{-64} which is a very small number. So, the quotient STS is generated with the same fingerprint collision probability and without exploding the memory.

E. Translating a STS into a Labelled Transition System

In order to use existing tools to check properties on a STS, we transform the quotient STS into a Labelled Transition System (LTS), that we call quotient LTS.

Definition 3: A LTS is 4-tuple $\mathcal{T} = \langle \mathcal{Q}, \mathcal{L}, \delta, s_0 \rangle$, where:

- $\mathcal{Q}$ is the set of states,
- $\mathcal{L}$ is the set of action labels,
- δ is the transition relation (a subset of $\mathcal{Q} \times \mathcal{L} \times \mathcal{Q}$),
- and s_0 is the initial state.

A transition (s_1, l, s_2) of δ, indicates that the system can move from state s_1 to state s_2 by performing action labelled by l.

c) Property preservation: The equivalence between checking a property given in $\text{LTL}_{\setminus x}$ (Linear Temporal Logic without the "next operator") on the quotient LTS and checking it on the original LTS is ensured by the preservation.

Proposition 1: Let φ be an $\text{LTL}_{\setminus x}$ formula, let $\mathcal{T}_\Phi$ and $\mathcal{T}_\Psi$ be two STSs such that $\mathcal{T}_\Psi \Rightarrow \mathcal{T}_\Phi$. If $\mathcal{T}_\Phi \models \varphi$ then $\mathcal{T}_\Psi \models \varphi$.

F. Usefulness of the Quotient LTS

The quotient LTS abstracts away the details of the concrete specification. Its main advantage is its small size. As properties are preserved between the concrete specification and its corresponding quotient STS, model checking properties can be done on the quotient LTS directly, which is a simple task. The quotient LTS is generated once and can be used to verify different properties (modulo the refinement mapping).

To express and check properties on the quotient STS, we use the CADP [6] toolkit. For this, we first adapt the label

 ISBN: 978-1-61208-438-1

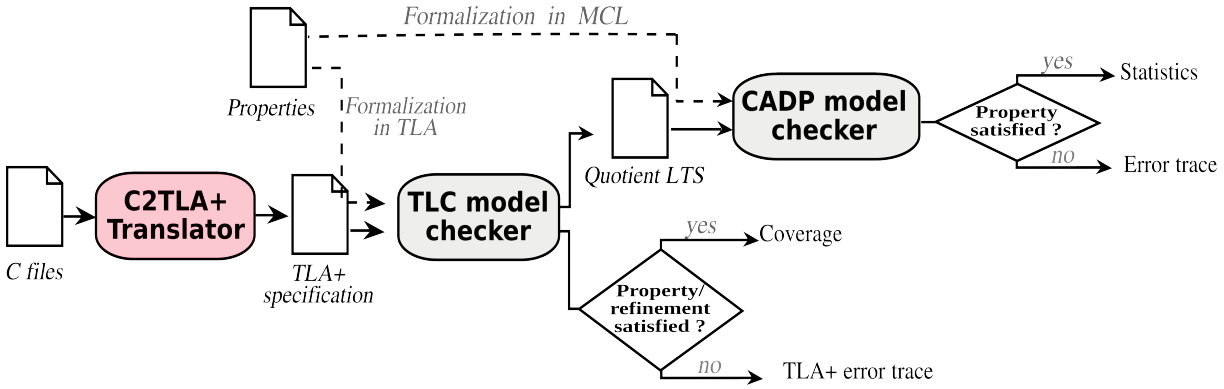


Figure 4. Verification flow of C programs

names such that LTS can be parsed by the CADP tools. Then, we express properties in the Model Checking Language (MCL) [7] language, the property specification language of CADP that can be verified by its associated model checker.

IV. Application of C programs

In this section, we implement the concept of refinement between TLA+ specifications and the quotient LTS on C programs. Figure 4 illustrates the verification flow of C programs. We use our tool C2TLA+ [8] to translate C programs into (a concrete) TLA+ specification. This latter can be checked directly against a set of properties, or against an abstract specification by defining the refinement mapping and the implementation relation between the concrete and the abstract specifications. Properties can be expressed in TLA to be verified using the TLC model checker. The quotient LTS is generated, and MCL properties can be verified by the CADP model checker.

In what follows, we briefly present how we specify the semantics of C in TLA+. We apply the described notions by considering the example of the dining philosophers. Finally, we assess the usefulness of using abstraction by giving results of properties verification using TLC and the CADP model checker.

A. TLA+ specification of a C program

C2TLA+ [8] generates a TLA+ specification that describes the behavior of the C program as a closed system according to a set of translation rules. A concurrent program consists in a set of C functions. In C2TLA+, concurrency is modeled by considering all possible interleaving of sequences of operations called *processes* (corresponding to threads in C). Each step of the complete specification is attributed to exactly one process. The C program is defined by a TLA formula in the form of (1). For more detailed information about the translation from C to TLA+, please refer to our previous work [8].

B. Illustrating Example

As an example, we consider the classic dining philosophers problem. One possible solution to this problem is the one that appears in Tanenbaum's popular operating systems textbook [9],, given in Figure 5.

In the implementation of this solution, the global semaphore `mutex` provides mutual exclusion for execution

```
#define N 4
#define THINKING 0
#define HUNGRY 1
#define EATING 2
#define LEFT(i) (i+N-1)%N
#define RIGHT(i) (i+1)%N
typedef int semaphore;
int state[N];
semaphore mutex;
semaphore sem[N];

void philosopher(int i)
{ while (1) {
   think();
   take_forks(i);
   eat();
   put_forks(i); }
}
void take_forks(int i) {
  P(&mutex);
  state[i] = HUNGRY;
  test(i);
  V(&mutex);
  P(&sem[i]);}

void put_forks(i)
{
  P(&mutex);
  state[i] = THINKING;
  test(LEFT(i));
  test(RIGHT(i));
  V(&mutex);
}

void test(i)
{
  if (state[i] = HUNGRY
      && state[LEFT(i)]!=
           EATING
      && state[RIGHT(i)]
          != EATING)
  {
    state[i] = EATING;
    V(&sem[i]);
  }
}
```

Figure 5. Tanenbaum's solution for the dining philosophers

of critical sections and the semaphore `sem[i]` ensures synchronization. The latters perform `P()` to acquire a lock and `V()` to release it, using "Compare-and-swap" primitive.

C. Refinement of Specifications

d) Abstract specification of the dining philosophers: We define a coarse-grained representation of the dining philosopher, illustrated by Figure 6 that captures the aspects of the system that interest us without giving all the details of its internal structure.

In order to check liveness properties, we consider that the philosopher cannot starve waiting for a fork, i.e., no philosopher is eating forever. This assumption is stated by the formula *Fairness*, where $\mathrm{WF}_{vars}(\mathcal{A})$ denotes weak fairness on action $\mathcal{A}$ and the symbol $\Diamond$ denotes the temporal operator *eventually*.

 ISBN: 978-1-61208-438-1

MODULE *Abstract_philosophers*

EXTENDS *Naturals*, *TLC*
CONSTANT N
VARIABLES *phil_state*, *forks*
$vars \triangleq \langle phil\_state, forks \rangle$
$fork\_available(i) \triangleq forks[i] = N$
$fork\_acquire(p, i) \triangleq forks' = [forks \text{ EXCEPT } ![p] = i]$
$forks\_release(p) \triangleq$
$forks' = [forks \text{ EXCEPT } ![p] = N,\ ![(p+1)\%N] = N]$
$fork\_release(p) \triangleq forks' = [forks \text{ EXCEPT } ![p] = N]$
$LEFT(p) \triangleq (p+1)$
$RIGHT(i) \triangleq \text{IF } (i = 0) \text{ THEN } (N-1) \text{ ELSE } (i-1)$

$think(ph) \triangleq$
$\wedge\ phil\_state[ph] =$ "think"
$\wedge\ fork\_available(LEFT(ph))$
$\wedge\ fork\_acquire((LEFT(ph),\ ph)$
$\wedge\ phil\_state' = [phil\_state \text{ EXCEPT } ![ph] =$ "hungry"]

$hungry(ph) \triangleq$
$\wedge\ phil\_state[ph] =$ "hungry"
$\wedge$ IF $(fork\_available(ph))$
THEN
$\wedge\ fork\_acquire(ph,\ ph)$
$\wedge\ phil\_state' = [phil\_state \text{ EXCEPT } ![ph] =$ "eat"]
ELSE
$\wedge\ fork\_release(LEFT(ph))$
$\wedge\ phil\_state' = [phil\_state \text{ EXCEPT } ![ph] =$ "think"]

$eat(ph) \triangleq \wedge\ phil\_state[ph] =$ "eat"
$\wedge\ forks\_release(ph)$
$\wedge\ phil\_state' = [phil\_state \text{ EXCEPT } ![ph] =$ "think"]

$Init \triangleq \wedge\ phil\_state = [i \in (0\ ..\ (N-1)) \mapsto$ "think"]
$\wedge\ forks = [i \in (0\ ..\ (N-1)) \mapsto N]$

$Spec \triangleq Init \wedge \Box[\exists\, i \in 0\ ..\ (N-1) :$
$think(ph) \vee hungry(ph) \vee eat(ph)]_{vars}$
$\wedge\ Fairness$

Figure 6. Abstract TLA+ version of the dining philosophers

$Fairness \triangleq$
$\wedge\ \forall\, i \in (0\ ..\ N-1) :\ \text{WF}_{vars}(hungry(i)) \wedge \text{WF}_{vars}(eat(i))$
$\wedge\ \forall\, i \in (0\ ..\ N-1) : \Box\Diamond(\text{ENABLED } \langle think(i) \rangle_{vars}) \implies (\Box\Diamond\langle eat(i) \rangle_{vars})$

e) Specifying the refinement relation: To check that the concrete specification generated by C2TLA+, implements the abstract version of the dining philosophers, we define the refinement relation as shown in Figure 7. In this section, we don't illustrate the translation of the C code, as the translation rules are described in our previous work [8].

The implementation relation is an implication formula $Spec \Rightarrow Abstract\_instance!Spec$.

D. Expressing properties

An interesting property that the implementation should hold is that the critical sections are protected with the primitives `P()` and `V()`. This property can be simply expressed in TLA+ (on the abstract specification) as follows:

MODULE *refinement_definition*

EXTENDS *Concrete_philosophers*
$philNum \triangleq load(\text{"unused"}, Addr\_N)$
$get\_val(addr, off) \triangleq$
$load(\text{"unused"}, [loc \mapsto addr.loc, offs \mapsto addr.offs + off]).val$

$refmap(addr) \triangleq$
$[i \in (0\ ..\ philNum) \mapsto$
LET $val \triangleq get\_val(addr\_state,\ i)$
IN IF $val = 0$ THEN "think"
ELSE IF $val = 1$ THEN "hungry"
ELSE "eat"]

$Abstract\_instance \triangleq$ INSTANCE *Asbtract_philosophers* WITH
$N \leftarrow philNum,$
$phil\_state \leftarrow refmap(Addr\_state)$

$Spec \implies Abstract\_instance\,!\,Spec$

Figure 7. Definition of refinement relation between abstract and concrete TLA+ specifications of the dining philosophers

$mutual\_exclusion \triangleq$
$\forall\, i \in (0\ ..\ (N-1)) : (phil\_state[i] =$ "eat") $\implies$
$(phil\_state[LEFT(i)] \neq$ "eat" $\wedge\ phil\_state[RIGHT(i)] \neq$ "eat")

The dining philosophers problem captures many aspects of liveness. Among liveness properties of the dining philosophers is starvation-freedom and deadlock freedom that we expressed in TLA+ as follows:

$NoStarvation \triangleq \forall\, i \in (0\ ..\ (N-1)) :$
$\Box((phil\_state[i] =$ "hungry") $\implies \Diamond(phil\_state[i] =$ "eat"))

$DeadlockFree \triangleq$
$\Box((\forall\, i \in (0\ ..\ (N-1)) : (phil\_state[i] =$ "hungry")) $\implies$
$(\forall\, i \in (0\ ..\ (N-1)) : \Diamond(phil\_state[i] =$ "eat")))

E. Verification results and comparison

We check that the concrete TLA+ specification (generated by C2TLA+) implements the abstract TLA+ specification (given in Figure 7). We also check the set of properties on these two specifications. We extract the quotient LTS from the concrete specification that we checked against the set of properties that we express in MCL. Table I shows the number of states and the verification time of the concrete and the abstract specifications using TLC, and the numbers of states, transitions and the time verification of the quotient LTS using CADP model checker. Experiments were carried on an Intel Core Pentium i7-2760QM with 8 cores (2.40GHz each) machine, with 8Gb of RAM memory. For 5 philosophers, the state space of the concrete TLA specification exceeds 113 millions states and its verification takes more than 10 hours to check the properties.

For the same number of philosophers, the abstract TLA specification generates 82 states and properties were checked in only 1 minute using TLC. On the other hand, the quotient LTS generated 47 states and its verification time is 42s. Due to the preservation properties, we can deduce that all the verified properties on the abstract TLA specification or on the quotient LTS are verified on the concrete specification. The use of

 ISBN: 978-1-61208-438-1

TABLE I. RUNTIMES OF MODEL CHECKING

Philos	Verification using TLC				Verification using CADP	
	Concrete Spec.		Abstract Spec.		Quotient LTS	
	States	Time(s)	States	Time(s)	States	Time(s)
3	395K	157	14	15	14	12
4	27.285K	1.080	32	23	20	20
5	113.285K	>36.000	82	64	47	42

abstraction reduces considerably the complexity of verification of C implementations.

When TLC reports that a transition violates the implementation formula $Spec \Rightarrow Abstract\_instance!Spec$, there is an error either in the concrete specification, the abstract specification, or the refinement mapping function. The trace given by TLC can help to determine which one of those is the case. We use our tool to translate this trace in C and get the C execution sequence that leads to the error.

V. Related work

Predicate abstraction [10] is a technique to abstract a program so that only the information about the given predicates are preserved. This technique is being used in SLAM [11], BLAST [12] and MAGIC [13]. Their approach has been shown to be very effective on specific application domains such as device drivers programming. SLAM uses symbolic algorithms, while BLAST is an on-the-fly reachability analysis tool. The Magic tool use LTS a specification formalism, and weak simulation as a notion of conformance of a system and its abstract specification.

These tools are applied to C programs and use automated theorem prover to construct the abstraction of the C program. The difficulty of these refinement-based approaches is that performing a refinement proofs between an abstract and a refined model require non trivial human effort and expertise in theorem proving to get the prover to discharge the proof obligations. SLAM cannot deal with concurrency, BLAST cannot handle recursion.

Besides predicate abstraction, several verification techniques for C programs have been proposed. CBMC [14] is a bounded model checker for ANSI C programs which translates a program into a propositional formula (in Static Single Assignment form), which is then fed to a SAT solver to check its satisfiability. CBMC explores program behavior exhaustively but only up to a given depth.

Compared to previous related works that use an over-approximation of the code implementation which is sound, our approach is based on constructing an executable abstract model, that can be expressed using TLA+ or by constructing the quotient LTS. Moreover, TLA+ is a logic that can express safety and liveness properties unlike SLAM, BLAST and CBMC which have limited support for concurrent properties as they only check safety properties.

VI. Conclusion and Future Work

We have defined an operational semantics of a TLA+ specification in terms of a STSs. We redefined the semantics of refinement between a high-level (concrete) and a low-level (abstract) TLA+ specifications using STSs and we illustrated a way to automatically construct a quotient STS from the concrete specification by extending the TLC model checker. We applied all these notions for verifying C programs. Experimental results show that verifying properties on the abstract model reduces considerably the complexity of the verification process.

As future work, we plan to extend this work on several interesting directions. We would like to generate TLA+ and MCL properties from the ACSL [15] specification language used in Frama-C. We envisage to benefit from Frama-C analysis of shared variables by several processes to generate TLA+ code with less interleaving between the processes, to reduce the state space. Finally, we aim to use the TLA+ proof system [16] to prove refinement between a concrete and abstract specifications.

References

[1] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled, Model checking. Cambridge, MA, USA: MIT Press, 1999.

[2] L. Lamport, Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, 2002.

[3] L. Leslie, "The Temporal Logic of Actions," ACM Trans. Program. Lang. Syst., vol. 16, no. 3, 1994, pp. 872–923.

[4] Z. Manna and A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems. New York, NY, USA: Springer-Verlag New York, Inc., 1992.

[5] M. Abadi and L. Lamport, "The Existence of Refinement Mappings," Theor. Comput. Sci., vol. 82, no. 2, 1991, pp. 253–284.

[6] H. Garavel, F. Lang, R. Mateescu, and W. Serwe, "CADP 2011: a toolbox for the construction and analysis of distributed processes," International Journal on Software Tools for Technology Transfer, vol. 15, no. 2, 2013, pp. 89–107.

[7] R. Mateescu and D. Thivolle, "A Model Checking Language for Concurrent Value-Passing Systems," in Proceedings of the 15th International Symposium on Formal Methods. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 148–164.

[8] A. Methni, M. Lemerre, B. Ben Hedia, S. Haddad, and K. Barkaoui, "Specifying and Verifying Concurrent C Programs with TLA+," in Formal Techniques for Safety-Critical Systems, C. Artho and P. C. lveczky, Eds. Springer, 2015, vol. 476, pp. 206–222.

[9] A. S. Tanenbaum, Modern Operating Systems, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.

[10] S. Graf and H. Saïdi, "Construction of Abstract State Graphs with PVS," in Proceedings of the 9th International Conference on Computer Aided Verification. London, UK, UK: Springer-Verlag, 1997, pp. 72–83.

[11] T. Ball, R. Majumdar, T. Millstein, and S. K. Rajamani, "Automatic predicate abstraction of c programs," in Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation, ser. PLDI '01. New York, USA: ACM, 2001, pp. 203–213. [Online]. Available: http://doi.acm.org/10.1145/378795.378846

[12] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre, "Software Verification with BLAST." Springer, 2003, pp. 235–239.

[13] S. Chaki, E. M. Clarke, A. Groce, S. Jha, and H. Veith, "Modular Verification of Software Components in C," IEEE Trans. Software Eng., vol. 30, no. 6, 2004, pp. 388–402.

[14] E. Clarke, D. Kroening, and F. Lerda, "A Tool for Checking ANSI-C Programs," in TACAS, K. Jensen and A. Podelski, Eds., vol. 2988. Springer, 2004, pp. 168–176.

[15] P. Baudin, J.-C. Filliâtre, C. Marché, B. Monate, Y. Moy, and V. Prevosto, ACSL: ANSI/ISO C Specification Language, version 1.4, 2009, [retrieved: October, 2015].

[16] D. Cousineau, D. Doligez, L. Lamport, S. Merz, D. Ricketts, and H. Vanzetto, "TLA+ Proofs," in 18th International Symposium on Formal Methods - FM 2012, D. Giannakopoulou and D. Méry, Eds., vol. 7436. Paris, France: Springer, 2012, pp. 147–154.

 ISBN: 978-1-61208-438-1

Revisiting The Package-level Cohesion Approaches

Waleed Albattah
Information Technology Department
Qassim University
Qassim, Saudi Arabia
e-mail: w.albattah@qu.edu.sa

Suliman Alsuhibany
Computer Science Department
Qassim University
Qassim, Saudi Arabia
e-mail: salsuhibany@qu.edu.sa

***Abstract*—Software measurements play a critical role in assessing software properties. Cohesion is one of the software properties that are considered to have a relationship with software quality. Many cohesion metrics have been proposed by researchers to assess cohesion on different software abstractions, i.e., class-level and package-level. The proposed package-level cohesion metrics in the literature seem to differ in their assessment of cohesion. In this paper, we try to investigate this issue and establish whether cohesion has only one concept. The conclusion of this paper encourages further investigation and comparison between the existing package-level cohesion metrics.**

Keywords—Cohesion; package; metric; measurement; software.

I. INTRODUCTION

With the increased importance of software measurements in assessing software properties, research works have produced and are continuing to produce new software measures. One specific type of measure is cohesion. Cohesion refers to the degree to which the elements of a specific component belong together [3].

During software maintenance, developers spend at least 50% of their time analysing and understanding software [2]. In object-oriented programming languages, e.g., Java, assembling only closely related classes into packages can improve software maintenance. Package cohesion metrics measure the coherence of a package amongst its elements that should be closely related. Cohesion is an internal attribute of software that affects its maintainability and reusability. Following the design principles [21], a high level of cohesion has as its goal to achieve software maintainability and promote its reusability [22][26].

Package-level cohesion research has received very little focus compared with research on other abstractions, e.g., class-level. When one examines the literature on package cohesion metrics, it is clear that there are significant differences in these metrics. Thus, the following natural question arises: do these metrics measure the same thing? This question will be addressed in this paper.

The paper is organised as follows. In Section II, we present Package Cohesion Principles [21]. The existing approaches to package cohesion are presented in Section III. Section IV presents the general example for all the existing approaches. The conclusion and future work are given in Section V.

II. PACKAGE COHESION PRINCIPLES

R. C. Martin [21] has presented six principles for package design, which have since become well-known and well-accepted. The first three principles are for package cohesion and they help to allocate system classes to packages. This allocation can help to manage the software during its development. In our previous work [23], the three package cohesion principles of Martin [21] were discussed and they are introduced here briefly from [23]:

1) The Reuse-Release Equivalence Principle (REP)
"The granule of reuse is the granule of release"

This states that the reuse of the code should be the same size as the release one. If a person decides to reuse someone else's code, he needs a guarantee that the support will continue and the release of new versions will be on the same original size. To ensure the reusability of the code, the author must organise the classes into reusable packages and then track them with the release.

2) The Common Reuse Principle (CRP)
"The classes in a package are reused together. If you reuse one of the classes in a package, you reuse them all"

This principle tells us which classes should be grouped together. As it states, the classes that tend to be reused together should be in the same package. It is more likely for reusable classes to depend on each other, so classes are rarely reused in separation. CRP states that the classes of a package should be inseparable, which means that if a package depends on this package, it should depend on all of its classes and not on a number of them. In short, classes that are not tightly coupled to each other should not be kept in the same package.

3) The Common Closure Principle (CCP)
"The classes in a package should be closed together against the same kinds of changes. A change that affects a package affects all the classes in that package and no other packages"

From the maintenance point of view, while the change is not avoidable, it should be controlled (minimised). If a change has been made on one package, there is no need to

 ISBN: 978-1-61208-438-1

re-release or revalidate packages that do not depend on the changed package. The CCP states that the classes in the package should not have different reasons to change.

While the previous two principles, REP and CRP, focus on reusability, the CCP focuses on the system maintainability. If a change is made on the code, it would be better to be on one package or on a few packages rather than being on many packages. The classes that are tightly related will change together. Hence, if they are kept in the same package, only one package or a small number of packages are going to be affected when a change happens. Also, the effort regarding revalidating and re-releasing of software will be minimised.

III. THE EXISTING PACKAGE COHESION APPROACHES

A number of cohesion approaches have been proposed on class and method levels [1][3]-[6]-[18]. In this section, we present some of the existing package-level cohesion approaches. A brief description is given for each. In the literature, Misic [19], Ponisio and Nierstrasz [22], Martin [21], Xu et al. [20], Zhou et al. [24], Abdeen et al. [25], and Albattah and Melton [23] have each proposed different methods to measure package cohesion. Each proposes a cohesion metric on the package level. A brief discussion for each approach is given next.

A. Approach by Misic

Misic [19] proposes a way to measure a functional cohesion. Since a number of approaches were focusing on cohesion as an internal structure issue, Misic claimed that cohesion could be also observed externally by focusing on its functional property regardless of the package's internal structure.

The approach measures the similarity of package objects (elements). The similarity between elements can be measured by looking at the external clients' usage patterns.

Method

Misic defined write and read range concepts. The write range of an object *O*, *W(O)*, refers to the set of objects (servers) used by this object (client). The read range of an object *O*, *R(O)*, refers to the set of objects (clients) used by this object (server).

Given a set of objects *S*, let *R(S)* be its client set (Read range), S_w the subset that IS? used to write its clients, and let $S_w(x)$ be the part of that subset that IS? used to write the client *x*. Then, the coherence is given by the following formula:

$$\psi(S) = \frac{\sum_{x \in R(S)} (\#S_w(x) - 1)}{\sum_{x \in R(S)} (\#S - 1)} \quad (1)$$

where

#*S* stands for the number of elements in *S*.

The coherence measure proposed by Misic can be calculated internally or externally. For internal coherence, the summation in the numerator and denominator will be restricted only for clients inside the questioned set. Similarly, the summation will be restricted only for clients outside the questioned set to measure the external cohesion.

B. Approach by Ponisio and Nierstrasz

Ponisio and Nierstrasz [22] proposed a similar approach to measure package cohesion. The proposed contextual metric measures the cohesion based on the common use by clients. The approach idea is to propose the Common-Use (*CU*) metric that measures the package cohesion by taking into account the way that a package's classes are accessed by other packages.

Method

CU measures the cohesion of package *P* by considering the use of its elements by the package clients. If all the clients use the same set of *P's* elements, these elements share the same responsibilities of *P*, and then *P* is cohesive. Instead, if the clients use a different set of *P's* elements, these elements have different responsibilities, and then *P* is not cohesive.

There is a need for weight to differentiate between client packages. Not all clients have the same degree in assessing *P's* cohesion. The weight reduces the influence of *P's* cohesion from the promiscuous clients.

Definition: "*We define the weight of a (client) package* P_{client} *as the inverse of the number of connections that* P_{client} *has with other packages."*

$$w(P_{client}) = \frac{1}{fan\ in(P_{client}) + fan\ out(P_{client})} \quad (2)$$

The definition of *CU* is given as follows:

"*We define Common-Use (CU) as the sum of weighted pairs of classes from the interface of a package having a common client package (f), divided by the number of pairs that can be formed with all classes in the interface.*"

$$\mathsf{CU} = \sum_{a,b \in I} \frac{\mathsf{f}(a,b) * weight(a,b)}{\#Pairs} \quad (3)$$

Where

$$\begin{aligned} I &= \mathsf{interface}(P) \\ \#Pairs &= \tfrac{|I| \times (|I|-1)}{2} \\ C &= \mathsf{clients}(a) \cap \mathsf{clients}(b) \\ \mathsf{f}(a,b) &= \begin{cases} 1, & if\ C \neq \emptyset \\ 0, & otherwise \end{cases} \\ weight(a,b) &= \textstyle\sum_{c \in C} \frac{w(c)}{|C|} \end{aligned}$$

The value of *CU* is between 0, which represents that the interface classes of the package have disjoint responsibilities, and 1, which means that the interface classes of the package are used together.

C. Approach by Martin

Martin [21] presents a set of principles of object-oriented package design. Three of these principles, package cohesion principles, try to help the software architect to organise classes over packages. These principles are: REP, CCP, and CRP, discussed earlier in Section II. The three principles aim to provide a high quality of package cohesion.

Method

Martin [21] proposed a number of simple package-level metrics. One of them is a relational cohesion of a package. The package cohesion metric is presented as an average number of internal relations per class. Regardless of the package external dependencies that are considered in other cohesion metrics, the metric measures the connectivity between package elements. This metric is quite simple to apply, and is given by:

$$H=(R+1)/N \quad (4)$$

where

H: package cohesion
R: number of internal relations
N: number of the package classes

The extra "1" in the numerator prevents cohesion *H* from equalling zero when *N*=1. This metric gives all internal relations the same weight and disregards the external ones. It has been applied to a number of software projects and is widely accepted.

D. Approach by Xu et al.

Xu et al. [20] propose an approach to measuring the package cohesion in Ada95 object-oriented programming language. The proposed metric is based on dependence analysis between package entities. It is assumed that the package may have objects and sub-programs.

Method

The package dependence graph (*PGDG*) describes all types of dependencies: inter-object dependence graph (*OOG*), inter-subprogram dependence graph (*PPG*), and subprogram-object dependence graph (*POG*). The method measures package cohesion according to *PGDG*. It assumes that package *PG* has *n* objects and m subprograms, where *n*, $m > 0$.

To present the measure in a unified model, a power for each object *PW(O)* is given:

$$\begin{cases} Cohesion(O) & O \text{ is a package object} \\ Cohesioin(PG(O)) & O \text{ is a type object} \\ 1 & others \end{cases}$$

Xu et al. [20] claimed that, according to the definitions, it is easy to prove that the measure satisfies the four properties given by Briand et al. [3][27] to develop a good cohesion measure.

However, an Ada package represents a logical grouping of declarations. The role of an Ada package is similar to the role of class in other languages, such as Java [24]. Thus, this package cohesion metric cannot be applied to the general example in the next section. An Ada package actually falls in the category of class-level cohesion metric.

E. Approach by Zhou et al.

Zhou et al. [24] proposed an approach to measuring package semantic cohesion called the Similar Context Cohesiveness (*SCC*). In this approach, the common context is used to assess the degree of relation between two components. *SCC* measures the inter- and intra-package dependencies that can reveal semantic cohesion between components.

Method

The proposed package cohesion measure *SCC* is based on the component context. The context of component *c* is composed of two sets: the components that depend on *c* and those that *c* depends on. The *SCC* metric is given by:

$$SCC(p) = \begin{cases} \dfrac{\sum_{(c_i,c_j)\in E(p)} Wgt(c_i,c_j)}{m(m-1)} & if \quad m>1 \\ 1 & if \quad m=1 \end{cases} \quad (5)$$

where

m: number of components *c* in *p*

$\mathrm{Wgt}(c_1, c_2) = \mathrm{CCS}(c_1, c_2) + \mathrm{Dep}(c_1, c_2)$

$$Dep(c_1,c_2) = \begin{cases} 1 & if \quad c_1 \xrightarrow{d} c_2 \quad or \quad c_2 \xrightarrow{d} c_1 \\ 0 & else \end{cases}$$

$CCS(c_1,c_2)$: denotes the similarity between the contexts of two components c_1 and c_2 , and is given by:

$CCS(c_1, c_2) = kRSS(c_1, c_2) + (1\text{-}k)DSS(c_1, c_2)$

k: represents the position's importance

RSS(c1, *c2*): similarity between $S_R(c1)$ and $S_R(c2)$

DSS(c1, *c2*): similarity between $S_D(c1)$ and $S_D(c2)$

$S_R(c) = \{c_i \mid c_i \xrightarrow{d+} c\}$

$S_D(c) = \{c_i \mid c \xrightarrow{d+} c_i\}$

F. Approach by Abdeen et al.

The approach proposed by Abdeen et al. [25] is based on the Simulated Annealing technique. The approach aims to reduce package coupling and cycles by moving classes between packages. Two metrics have been defined for this purpose, coupling and cohesion metrics.

Method

The approach automatically reduces package coupling and cycles by moving classes between packages considering the existing class organisation and package structure. This approach can help maintainers to define: the maximum number of classes that can change their packages, the maximum number of classes that a package can contain, and

 ISBN: 978-1-61208-438-1

the classes that should not change their packages and/or the packages that should not be changed. A set of measures is defined to determine and quantify the quality of a package. The number of package dependencies ($|P_D|$) normalises these measures.

The package cohesion metric is defined to be the direct dependencies between its classes. Hence, the cohesion of a package P is proportional to the number of its internal dependencies ($|P_{Int.D}|$) according to the CCP Principle [19]. The cohesion quality is given as follows:

$$CohesionQ(p) = \frac{|p_{Int.D}|}{|p_D|} \quad (6)$$

where

$|P_D|$ is the number of all internal and external dependencies of classes in the package.

G. Approach by Bauer and Trifu

Bauer and Trifu [28] have proposed an approach, architecture-aware adaptive clustering, to produce meaningful decompositions in a system. They have evaluated their approach by defining two metrics: the average cohesion of a subsystem and the average coupling between subsystems.

Method

The approach was based on providing better understanding of the system. They tried to recover from the original decomposition and then impose an appropriate structure. The new structure aims to maximise subsystems cohesion. To evaluate the recovered subsystem decomposition, they performed a comparative study that is based on two criteria, accuracy and optimality. For accuracy, they compared the resulting decompositions with both the original package structure and the ideal Common Reuse Principle structure of [21]. For optimality, they used some optimality metrics to show whether the resulting decompositions have high cohesion and low coupling. To evaluate their approach, two metrics were defined: average cohesion of the subsystems and average coupling between the subsystems of a given decomposition. The average cohesion metric is given by:

$$avgCohesion(D) = \frac{\sum_{\substack{S_i \in D \\ |S_i|>1}} \frac{noInternalEdges(S_i)}{\frac{|S_i|^2-|S_i|}{2}}}{|D|^*} \quad (8)$$

where

D: a composition

noInternalEdges(Si): number of edges between classes in *Si*

$|D|^*$: number of subsystems except single-class subsystems in D

Si: subsystem number *i* in *D*

$|Si|$: number of classes in subsystem *Si*

H. Approach by Seng et al.

The approach by Seng et al. [29] aims to develop existing object-oriented system decompositions by defining new decompositions with better metric values and fewer violations of design principles. They defined the problem as a search problem. The quality of the resulting subsystem decompositions is measured by the fitness function that combines software metrics and design heuristics.

Method

The fitness function consists of cohesion, coupling, complexity metrics, as well as cyclic dependencies and bottleneck heuristics. The value of each individual function is between 0 and 1, where the optimal value is 1. The cohesion of a system s is the summation of cohesion values for the individual subsystems in s. The cohesion for a subsystem s_i is measured by counting the number of different classes in s_i known by some class $c_j \subset s_i$,(#k(c_j)), and dividing this by the square number of classes in s_i , (#c(s_i)). The resulting value can be normalised if divided by the number of subsystems (#s).

$$cohesion(s) \quad : \quad \frac{\sum_{i=1}^{\#s} \sum_{j=1}^{\#c(s_i)} \frac{\#k(c_j)}{\#c(s_i)^2}}{\#s} \quad (7)$$

I. Approach by Tagoug

Tagoug [30] has proposed coupling and cohesion metrics on subjects, which are similar to packages. Each subject is a collection of classes. The approach aims to measure cohesion and coupling at the system level. The quality metric, which combines cohesion and coupling values, measures the decomposition's quality as early as the analysis and design phases of the software development lifecycle.

Method

The two metrics measure the quality of object-oriented decomposition. The cohesion metric focuses on the interactions of components inside a subject, while the coupling metric focuses on the interactions of components among subjects. The cohesion of subject E is given by:

$$C(E) = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Wij}{W_{\max} * (n*(n-1)/2)} \quad (9)$$

where

E: a set of classes of S.
Wij: the sum of the weights of links in *Lij*.
Lij: the set of all links between classes *Pi* and *Pj*.
Wmax = max $\{Wij\}$ in system S

$n = |E|, n > 1$

The cohesion value is between 0, i.e., there are no links between classes, and 1, maximum links with maximum weight. The weights of links between classes of a subject are ordered in Table I based on the degree of association according to the object-oriented expert designers.

TABLE I. WEIGHTS OF LINKS BETWEEN CLASSES.

Links Type	Weights (*Wij*)
Whole Part Structure	0.9
Inheritance	0.8
Instance Connection	0.7
Message Connection	0.6
Conceptual Link	0.5

J. Approach by Albattah and Melton

The approach by Albattah and Melton [23] is motivated by the package cohesion principles [21]. They proposed two different cohesion metrics to measure two different cohesion concepts or types based on Martin's package cohesion principles in [21]. The first cohesion type, Common Reuse (*CR*), includes the factors that help in assessing *CR* cohesion. Similarly, the second cohesion type, Common Closure (*CC*), includes the factors that help in assessing *CC* cohesion. After each type of cohesion is measured by itself, the two values of *CR* and *CC* may be combined to one unified value of package cohesion, while still recognising the two types.

Method

The *CR* metric measures cohesion based only on the common reuse factors of the package. The elements of a package have different degrees of reachability. Reachability of a class in a package is the number of classes in the same package that can be reached directly or indirectly. The *CR* metric is defined as follows:

"Let $c \in C$, and suppose there is an incoming relation to *c* from a class in a different package. Then *c* is called an in-interface class. The cardinality of the intersection of the hub sets of all the in-interface classes in *C* divided by the number of classes in *C* is the *CR* of *P* ".

$$CR = |\cap_{\text{In-interface class hub sets}}| / |C| \quad (10)$$

where

Hubness(c) = {d ∈ C: if there is a path c → d}

C: set of classes in package *P*

c and d: classes in C

The *CC* metric considers the package dependencies on other packages as well as the internal dependencies between classes of the package. The classes of the package should depend on the same set of packages and, thus, they will have the same reasons for a change. The *CC* metric is defined as follows:

"The cardinality of the intersection of the reachable sets divided by the cardinality of the union of the sets represents the *CC* of *P* ".

$$CC = (|\cap_{\text{Reachable Package sets}}| / |\cup_{\text{Reachable Package sets}}|) \quad (11)$$

The combined cohesion *CH* is defined as follows:

$$CH = \frac{\sqrt{2} - D}{\sqrt{2}} \quad (12)$$

$$D = \sqrt{(1-CR)^2 + (1-CC)^2} \quad (13)$$

IV. THE GENERAL EXAMPLE

While we try to understand each of the previously presented approaches, we rely on our best understanding for each. One method of empirical investigation is to apply all the approaches on the same situation and compare the results. The approaches have been applied to measure the cohesion of *P1* in Figure 1. The concern is to measure the cohesion of *P1* only for the purpose of comparison between the approaches. If all the approaches rely on the same idea, their assessments of the cohesion of *P1* will be alike. Otherwise, they probably rely on different concepts of package cohesion.

In Figure 1, there are six packages and a number of classes in each package. The arrows represent the dependencies between classes within the same package, i.e., in *P1*, or between classes in different packages. The direction of the dependency is very important because it shows the depended-upon class. For example, *C6* depends on *C2* but not the opposite. In the figure, *P1* has four classes that have incoming and outgoing dependencies. Using the presented approaches, we try to measure how cohesive are the classes of *P1*. It is worth mentioning that all the presented approaches consider the dependencies between classes to measure cohesion, but in different ways. Some approaches, such as Albattah and Melton [23], consider the direction of the dependencies. However, some other approaches, such as Martin [21], do not consider the direction of the dependencies. For this difference and other differences between the presented approaches, it is expected to find distinct cohesion assessment values for *P1*.

Again, all calculations are made based on our own understanding of each approach.

 ISBN: 978-1-61208-438-1

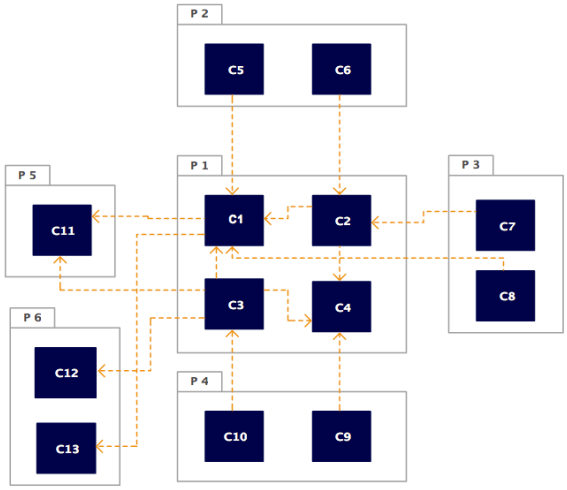


Figure 1. The general example.

Table II presents the cohesion values of package *P1* for the different approaches.

TABLE II. COHESION VALUES OF THE PRESENTED APPROACHES.

Approach	Cohesion			
	Metric	Value	Min	Max
Misic [19]	*Ψ(S)*	0.33	0	1
Ponisio and Nierstrasz [22]	*CU*	0.125	0	1
Martin [21]	*H*	1.25	> 0	N(N-1)*
Zhou et al. [24]	*SCC(p)*	0.36	0	1
Abdeen et al. [25]	*CohesionQ(p)*	0.29	0	1
Bauer and Trifu [28]	*avgCohesion(D)*	0.67	0	1
Seng et al. [29]	*cohesion(s)*	0.25	0	1
Tagoug [30]	*C(E)*	0.67 **	0	1
Albattah and Melton [23]	*CH*	0	0	1

* N: number of classes in the package

**Assuming that all the connections are instance connections with 0.7 weights.

Although all the presented approaches have the same range of cohesion values except Martin's approach [21], they end up with different cohesion values for the same package, i.e., *P1* in Figure 1. For example, the approaches by Bauer and Trifu [28] and Tagoug [30] assess the cohesion of *P1* as relatively high. In contrast, the approach by Albattah and Melton [23] assesses the cohesion of *P1* as poor.

This simple comparison raises a question about the theory behind these different approaches. The distinct evaluation results for the same package means that the presented approaches rely on different views of cohesion. These views can be noticed by investigating the presented approaches. We believe cohesion has different types or parts and some approaches focus only on one part. This can lead to misleading cohesion assessments. Cohesion has three different concepts that led to different approaches. The first concept considers cohesion as an internal property of a package that can be measured from inside the package only, such as the approach by Martin [21]. The second concept considers cohesion as a property that can be measured from outside the package, such as the approach by Ponisio and Nierstrasz [22]. The third concept considers cohesion to be measured from both inside and outside the package, such as the approach by Albattah and Melton [23].

These three concepts represent three scopes where cohesion has been measured in the presented approaches. The scope of package cohesion can be used to classify the presented approaches. Table III presents this classification based on the scope of cohesion used in each approach, i.e., internal, external, or both.

TABLE III. CLASSIFICATION OF THE PRESENTED APPROACHES.

Approach	Method	Scope of Cohesion	
		Internal	External
Misic [19]	External Objective		✓
Ponisio and Nierstrasz [22]	Common Use of the package		✓
Martin [21]	Relational Cohesion	✓	
Zhou et al. [24]	Similar Context Cohesiveness	✓	✓
Abdeen et al. [25]	Dependency Analysis	✓	
Bauer and Trifu [28]	Average Cohesion	✓	
Seng et al. [29]	Dependency Analysis	✓	
Tagoug [30]	Interactions inside the package	✓	
Albattah and Melton [23]	Common Reuse & Common Closure	✓	✓

The classification in Table III can reveal, somehow, the reason behind the diversity of package cohesion approaches that led to distinct results in Table II. Package cohesion has been viewed in different ways. It is worth saying that all the views may be right but they are different. This leads to the idea that there is more than one type of cohesion. The previous research works treated cohesion as one type or one concept, except for the research carried out by Albattah and Melton [23], and this was not accurate in our opinion.

We support the idea of Albattah and Melton [23] that is presented in this paper about cohesion. They defined cohesion as an internal property of the package and it has two different types. The first type can be measured from outside the package and it represents how well the classes in

 ISBN: 978-1-61208-438-1

the package cooperate to provide a service to classes outside the package. The second type measures how well the classes in the package are closed in using classes in other packages. This type represents the closure of the package's classes against the same kind of changes, which is the same set of depended-upon packages.

We believe cohesion is affected by internal and external factors and it should be treated based on this concept for accurate assessments. On the other hand, the generalised term of "cohesion" should not be used if the approach only relies on one consideration, i.e., internal or external. Terms such as "Common Closure Cohesion" and "Common Reuse Cohesion" can be used to describe the approach that relies on one consideration, i.e., internal and external, respectively. It is worth saying that Martin [21] has established a theory behind the internal and external factors by presenting the three package cohesion principles already discussed in Section II. Moreover, Martin's cohesion principles have been used to distinguish between package cohesion types in our previous work, Albattah and Melton [23].

V. Conclusion and Future Work

In this paper, a preliminary research survey on package cohesion approaches is presented. The survey shows that there is a rich variety of package cohesion understanding, which has led to the production of different package cohesion metrics in which each of them is based on a specific view of cohesion. We believe that there are significant differences in these metrics. Thus, the metrics of these approaches measure different things. The example given in the paper shows different values of cohesion and motivates us to classify the presented approaches. A preliminary classification reveals the reason behind the diversity of package cohesion approaches that led to distinct results in the given example. Obviously, the scope of cohesion is the foundation for this classification. We conclude that cohesion is more than one part and the term of "cohesion" should not be used unless the internal and external considerations are taken into account. Otherwise, terms such as "Common Closure Cohesion" and "Common Reuse Cohesion" can be used to describe the approach that relies on one consideration, i.e., internal and external, respectively.

In future work, we plan to examine the role of package cohesion in predicting software maintainability and software reusability.

References

[1] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design." IEEE Transactions on Software Engineering, 20.6 (1994): 476-493.

[2] V. Basili, "Evolving and packaging reading technologies." Journal of Systems and Software 38.1 (1997): 3-12.

[3] L. Briand, J. Daly, and Jürgen Wüst, "A unified framework for cohesion measurement in object-oriented systems." Empirical Software Engineering 3.1 (1998): 65-117.

[4] L. Briand, S. Morasca, and V. Basili, "Measuring and assessing maintainability at the end of high level design." Conference on Software Maintenance Proceedings, 1993. CSM-93 (pp. 88-87), IEEE, 1993.

[5] B. Henderson-Sellers, L. Constantine, and I. Graham, "Coupling and cohesion (towards a valid metrics suite for object-oriented analysis and design)." Object Oriented Systems 3.3 (1996): 143-158.

[6] S. Orlov and A. Vishnyakov, "Metric Suite Selection Methods for Software Development of Logistics and Transport Systems." Proceedings of the 11th International Conference "Reliability and Statistics in Transportation and Communication" (RelStat'11), 19-22 October 2011, Riga, Lativia, p.301-310.

[7] J. Eder, G. Kappel, and M. Schrefl, "Coupling and cohesion in object-oriented systems." Technical Reprot, University of Klagenfurt, Austria (1994).

[8] Y. Lee, B. Liang, S. Wu, and F. Wang, "Measuring the coupling and cohesion of an object-oriented program based on information flow." In Proc. International Conference on Software Quality, Maribor, Slovenia, 1995, (pp. 81-90).

[9] G. Gui and P. Scott, "Coupling and cohesion measures for evaluation of component reusability." Proceedings of the 2006 International workshop on Mining software repositories, 2006, (pp. 18-21). ACM, 2006.

[10] M. Hitz, and B. Montazeri, "Measuring coupling and cohesion in object-oriented systems." Proceedings of the International Symposium on Applied Corporate Computing. Vol. 50. 1995.

[11] W. Li and S. Henry, "Maintenance metrics for the object oriented paradigm." Proceedings of First International Software Metrics Symposium, 1993, (pp. 52-60), IEEE, 1993.

[12] S. Chidamber and C. Kemerer, "Towards a metrics suite for object oriented design." Vol. 26. No. 11 , 1991, (pp. 197-211). ACM.

[13] J. Bieman and Byung-Kyoo Kang, "Cohesion and reuse in an object-oriented system." ACM SIGSOFT Software Engineering Notes. Vol. 20. No. SI. ACM, 1995.

[14] J. Bieman and Linda M. Ott, "Measuring functional cohesion." IEEE Transactions on Software Engineering, Vol. 20, No. 8, (1994): (pp 644-657).

[15] L. Etzkorn, S. Gholston, J. Fortune, C. Stein, D. Utley, P. Farrington, and G. Cox, "A comparison of cohesion metrics for object-oriented systems." Information and Software Technology Vol. 46, No. 10, (2004): (pp 677-687).

[16] H. Chae, Y. Kwon, and Doo-Hwan Bae, "A cohesion measure for object-oriented classes." Software-Practice and Experience, Vol. 30, No.12, (2000): (pp 1405-1432).

[17] L. Ott, , J. Bieman, B. Kang, and B. Mehra, "Developing measures of class cohesion for object-oriented software." In Proc. Annual Oregon Workshop on Software Merics (AOWSM'95), vol. 11. 1995.

[18] J. Bansiya, L. Etzkorn, C. Davis, and W. Li, "A class cohesion metric for object-oriented designs." Journal of Object-Oriented Programming, Vol. 11, No. 8, (1999): (pp 47-52).

[19] V. Misic, "Cohesion is structural, coherence is functional: Different views, different measures." Proceedings of the Seventh International Software Metrics Symposium, 2001, (pp. 135-144), METRICS. IEEE, 2001.

[20] B. Xu, Z. Chen, and J. Zhao, "Measuring cohesion of packages in Ada95." ACM SIGAda Ada Letters, Vol. 24, No.1, (2004): (pp 62-67).

 ISBN: 978-1-61208-438-1

[21] R. C. Martin, Agile software development: principles, patterns, and practices. Prentice Hall PTR, 2003.

[22] L. Ponisio and O. Nierstrasz, “Using contextual information to assess package cohesion”, Technical Report No. IAM-06-002, 2006, Institute of Applied Mathematics and Computer Sciences, University of Berne, 2006.

[23] W. Albattah and A. Melton, “Package cohesion classification”, in: Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on, IEEE, 2014, (pp. 1–8).

[24] T. Zhou, B. Xu, L. Shi, Y. Zhou, and L. Chen, "Measuring package cohesion based on context." IEEE International Workshop in Semantic Computing and Systems, 2008. WSCS'08, (pp. 127-132), IEEE, 2008.

[25] H. Abdeen, S. Ducasse, H. Sahraoui, and I. Alloui, "Automatic package coupling and cycle minimization." 16th Working Conference on Reverse Engineering, 2009, (pp. 103-112), WCRE'09. IEEE, 2009.

[26] T. Biggerstaff and A. Perlis, "Software reusability: vol. 1, concepts and models." (1989).

[27] L. Briand, S. Morasca, and V. Basili, "Property-based software engineering measurement." IEEE Transactions on Software Engineering, Vol.22, No.1, (1996): (pp 68-86).

[28] M. Bauer and M. Trifu, "Architecture-aware adaptive clustering of OO systems." Eighth European Conference on Software Maintenance and Reengineering Proceedings 2004, CSMR 2004, (pp. 3-14), IEEE, 2004.

[29] O. Seng, M. Bauer, M. Biehl, and G. Pache, "Search-based improvement of subsystem decompositions." In Proceedings of the 7th annual conference on Genetic and evolutionary computation, 2005, (pp. 1045-1051), ACM, 2005.

[30] N. Tagoug, "Object-oriented system decomposition quality.", 7th IEEE International Symposium on High Assurance Systems Engineering Proceedings, 2002, (pp. 230-235), IEEE, 2002.

 ISBN: 978-1-61208-438-1

Towards a Technical Debt Management Framework based on Cost-Benefit Analysis

Muhammad Firdaus Harun, Horst Lichter
RWTH Aachen University, Research Group Software Construction
Aachen, Germany
e-mail: {firdaus.harun, horst.lichter}@swc.rwth-aachen.de

***Abstract*—Technical debt (TD) is a metaphor of bad software design or immature artifacts of a software system. The metaphor has been quite intensively researched especially on how to identify the TD symptoms, (e.g., system deficiencies or architecture violations) explicitly. Although the TD identification is quite important in the TD management process, a systematic management of TD and how to reduce it should also be considered important in each release of the development project. Otherwise, the software becomes more and more unmaintainable. In this paper, we introduce a framework to manage and reduce the TD of software systems. As it is based on quantification and a cost-benefit analysis, it is called Cost-Benefit based Technical Debt Management (CoBeTDM). CoBeTDM defines explicit phases focusing on the most important aspects of TD management: identification, monitoring, and prioritization. Overall, CoBeTDM should support managers to take the right decisions regarding the software evolution and the reduction of the collected TD at the right time.**

Keywords–technical debt management; code smells; architecture smells; refactoring; cost-benefit analysis.

I. Introduction and Motivation

It is a must to implement a payback strategy (when and how to determine to pay it back) to reduce technical debt for every software organization. It has been reported that TD exists in most of the software systems [1]. If we do not cautiously manage the debt or have no strategy to pay it back, the system may finally go to the "bankruptcy" phase, i.e., the software is unmaintainable and the maintenance cost will increase continuously. In general, refactoring is one of the strategies to pay it back. Refactoring has typically been used as a mean to improve detailed design and code quality. In this paper, refactoring will be referred to as an effort to improve existing software either on code or architecture-level without changing the behaviour of the system.

Commonly, project managers are always juggling on the decision making either to add new features or to make changes, (i.e., maintenance or refactoring) in a release cycle. It is always complicated to decide, which refactoring task should be done first or could be postponed. Therefore, quantification of refactorings should be implemented to identify, which effort can achieves maximum benefit and minimize risk. A simple cost-benefit analysis is a simple approach that could be applied to quantify it as introduced [2]. Borrowing from economic domain, a cost is a principal that indicate effort estimation to resolve a TD item and a benefit is an interest that indicate less probability impact to the software system. However, the quantification cannot answer the question "How the refactoring effort could be paid off to the identified TD, i.e., Return On Investment (ROI)?". ROI is a predictor that shows a particular refactoring may improve the design and save the maintenance cost in the future. Besides the unanswered question of ROI, it lacks of risk factors consideration and misses the payback strategy over releases. Therefore, to reduce technical debt and to sustain software quality in software development continuously, a wise decision making should be made based on a cost-benefit analysis.

In this paper, we want to introduce an approach of technical debt management based on cost-benefit analysis. The remainder of this paper is organized as follows: Section II presents the research goals. Section III describes our approach to Technical Debt Management and its phases. Section IV discusses relevant related work and Section V concludes the paper.

II. Goals

In order to support software development organizations to systematically manage the TD of their software systems, we propose an approach called *Cost-Benefit based Technical Debt Management* (CoBeTDM). Its overall goal is to provide a framework to manage and reduces TD based on cost-benefit analysis for each release. To achieve this main goal, the following sub-goals should be fulfilled:

1) Provide a **debt item model** (see Table I) that comprises all information of code and architecture smells and the effort needed to resolve them.
2) Quantify **cost and benefit** for each possible refactoring of a particular debt item. This enables to select the "best" refactoring based on the expected ROI.
3) Provide a **structured process** on how to strategically pay back the TD based on quantified cost-benefit of refactoring effort either tactically or proactively.
4) Develop a **toolbox** to support TD management and to monitor the identified debt items.

CoBeTDM defines four phases as shown in Figure 1 (see Section III for details):

1) **Identification & Assessment**: Here, the focus is to identify and measure the worst smells as well as to model them by means of debt items.
2) **Monitoring**: In order to know the development of TD and its trend, it has to be monitored continuously.
3) **Quantification & Prioritization**: Based on a cost-benefit analysis of each possible refactoring associated with a debt item, the quantified refactorings are prioritized based on their ROI.
4) **Repayment**: Selected refactorings will be inserted into backlog for current or later releases in order to reduce the TD.

 ISBN: 978-1-61208-438-1

III. COST-BENEFIT BASED TECHNICAL DEBT MANAGEMENT (COBETDM)

Relevant and accurate data is needed to quantify TD related cost and benefit for a software system. It is to support managers to take the right decisions. To provide this data, a collection of metrics that characterize code and architecture smells could be applied.

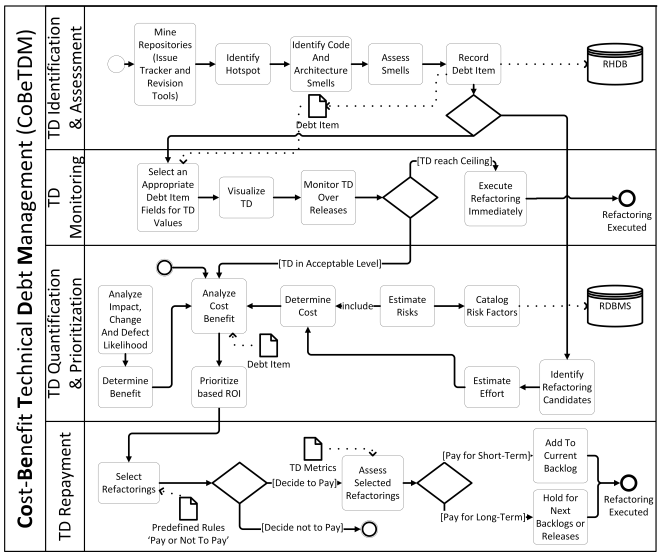


Figure 1. CoBeTDM Process

Modeling Debt Items. We propose a data structure (called *debt item*) to store all relevant and accurate information of all detected code and architecture smells. It will be stored in Release History Database (RHDB) - a database that stores data model that link between bug tracking system and versioning system. The data structure is depicted in Table I.

TABLE I. DEBT ITEM DATA STRUCTURE

Field	*Description*
Id	Unique identifier of debt item
Issue\Case	Task IDs or Case IDs, which represent a critical artifact (hotspot)
Dependency	Case IDs that depends on this debt item
Frequent Change	How many modifications have been made for one release?
Class	Class name
Code Smells	List of detected smells and its metrics values
Architecture-level	Architecture elements such as class, package, module or layer name
Architecture Smells	List of detected architecture smells and its metric values
Worst Smells	Sum of frequent change + code smells value + architecture smells value
Principal	Effort estimation to resolve this debt item
Interest	Extra effort estimation to resolve this debt item
Impact	Other artifacts that are impacted
When-to-Release	Release number
Responsible	A person or unit responsible for this debt item

A. TD Identification and Assessment

The identification of deficiencies of a software system is a must in the early phase of TD management. In CoBeTDM, the detection of bad smells is done in the following two steps: 1) **Hotspot detection**: Here, the goal is to find frequent changes, (i.e., unstable) software artifacts, which might be critical for the evolution of the system; 2) **Code and architecture bad smells detection**: For all identified hotspots, the worst code and architecture smells will be detected.

Hotspots detection. Hotspot detection is an approach to find the most critical artifacts of a software system. In this paper, the critical artifact means the module becomes unstable for certain releases, (i.e., frequent change over releases) and indicates strong increase in size and complexity (using metrics such as Lines of Code and McCabe Cyclomatic Complexity). It is important to detect the hotspot due to the symptom cost more than other code deficiencies. It is because we consistently have to pay back to tame it for every release. The hotspots detection can be supported by a dedicated mining repository approach where data from bug tracking and versioning tools are extracted, filtered and classified by tracking any frequent changes of contained artifacts. Currently, we manually map their IDs between Bugzilla and the Git repository. Then, we examine these artifacts by analyzing its size and complexity trend over releases. As a result, the artifacts that have many changes, (i.e., high maintenance activities) within the release could be detected as potential hotspots. We quantify criticality of an artifact by the number of changes that have been made, (i.e., Git log entries) performed for fixing bugs, (i.e., different severity levels of bugs) that were reported for specific releases. E.g., up to release 1, `CriticalPackage` of Application X got 200 modification from 130 bugs rated critical. Besides that, the identified artifact has a significant increase in size and complexity. From 4,000 LOC in release 0.9. increases to 10,000 LOC in release 1.0. Furthermore, the complexity of the package increases from 30 in release 0.9 to 50 in release 1.0. This symptom can be called as a critical artifact or hotspots.

Code Smells Detection. To detect code smells of the identified hotspots, we use a tool called *iPlasma* introduced by [3]. The tool shows a list of smells and its metric values. The highest metric values for each smell will be selected and prioritized. This data is recorded into a debt item to be used in the next phase. For instance, the `CriticalPackage` as detected as critical artifact previously will be assessed by *iPlasma*. The tool will detect any possible bad code smells. E.g., `CriticalPackage` contains `GodClass`, which has been detected as God class. The class has for example, 453 methods, defines 114 attributes and is more than 3500 lines long. It may also contain other smells, e.g., code duplication, data class etc., in this particular case, we focus on God class due to its refactor effort is quite high [4] compared to other smells. The tool will show relevant metrics for God class such as Access to Foreign Data (ATFD), Weighted Method Count (WMC) and Tight Class Cohesion (TCC). Each metrics value will be shown, e.g., as WMC (107), TCC (0.0) and ATFD (28). The metric values then will be recorded into *Code Smells* field in a debt item as shown in Table II.

Architecture Smells Detection. Next, the identified smells will be analyzed to detect architecture smells. The metrics introduced by [5] can be applied at class-, package- or subsystem-level. These smells can be detected by using existing tools such as Sonargraph-Architect [6]. The metric values produced by the architecture analysis tool will be stored as well into their respective debt items. In previous example, `CriticalPackage` was detected as critical artifact and contains `GodClass`. The class might contain cyclic dependency with other classes both within or outside the package. To detect the smells, the aforementioned tool can be used. For example, Sonargraph-Architect can detect it between classes or packages visually. It also displays the information regarding number of cycles and artifacts name. Then, cyclic value will be recorded into *Architecture Smells* field in the debt item.

 ISBN: 978-1-61208-438-1

TABLE II. DEBT ITEM EXAMPLE

Field	*Description*
Id	DI001
Issue\Case	#1234, #1235, #1236: Critical bugs of Application X
Dependency	#4321 #4322: Other critical bugs of Application X
Frequent Change	200 modification
Class	`GodClass`
Code Smells	God Class: WMC(107), TCC(0.0), ATDF(28)
Architecture-level	`CriticalPackage`
Architecture Smells	Cycle Dependency: Cyclic(10)
Worst Smells	200 + (107+0.0+28) + 10 = 345
Principal	16 hours (code smells) + 4 hours (cyclic dependency) = 20 hours: 1) *cost_to_split_a_class = 8 hours*. At least 2 classes will be partitioned for refactoring; It means 8×2; 2) *cost_to_cut_an_edge_between_two_files = 4 hours*. At least 2 files will be cut for refactoring. Both estimation based on [7]
Interest	2 hours, i.e., estimation extra work
Impact	15 classes and 2 packages
When-to-Release	Current: 1; Next: 1.1
Responsible	Mr. X

Assessing Bad Smells. After collecting the data from both code and architecture smells detection, we can analyze the obtained metric values to detect, which artifacts contain worst smells (i.e., highest in identified smells). For this means, we propose to apply the following formula *Worst Smells of Detected Critical Artifact = Most Frequent Changes + Highest Metrics of particular Code Smells + Highest Metrics of particular Architecture Smells*. See *Worst Smells* field in Table II. The worst smells value, then, will be compared with other debt items. The high value will be prioritized first instead of the low value. Besides that, the value could be used for TD monitoring as we explain in the next section.

B. TD Monitoring

To answer important questions such as: 1) How much TD do we have right now or in the current release?; 2) Is the TD at an acceptable level or not? 3) Does the TD continuously grows for each release?; 3) What is an acceptable **threshold** value of TD of each release? What is a maximum TD (**debt ceiling**) or minimum TD (**debt baseline**) for each release?; 4) How to react when the TD reaches the ceiling?; the TD has to be monitored continuously. Therefore, the TD data and its trend has to be visualized appropriately. The first idea is shown in Figure 2. Currently we are developing ideas and solutions for a systematic TD monitoring approach. Examples are: 1) A dedicated dashboard used to visualize TD data based on the managers' information needs. For example, the worst smells for certain release, high or low impact of debt item etc.; 2) A process to conduct semi-structured interview with managers or lead developers in order to gain information such as acceptable and minimum vs. maximum TD; 3) Development of a risk mitigation strategy framework that could be applied if TD reaches debt ceiling.

C. TD Quantification and Prioritization

In this phase, the debt items are quantified to perform a cost-benefit analysis. By cost, we mean the estimated effort and extra effort, (i.e., principal + interest) of a particular possible refactoring for a debt item. The cost value is stored together with the estimation risk, (i.e., judgment by experts) that may resulted from the refactoring. Then, it should be cataloged and stored in the database, (e.g., RDBMS) in order to be referred in the future. Then, the benefit is estimated, i.e., the less effort of refactoring, which gives positive impact. Currently, the benefits values are estimated based on the impact analysis

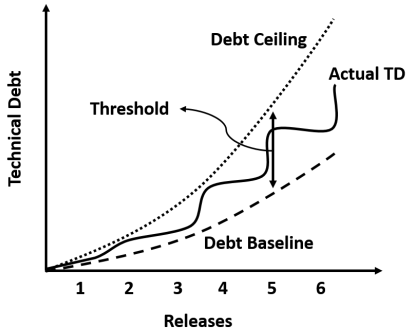


Figure 2. TD Trend over Releases

in particular dependency analysis. In addition, we also add defect and change likelihood as properties, while calculating benefits. The less value of both likelihood are potentially has less frequent of the same symptoms in the future. It means that the refactoring effort for maintenance and correction will decline.

Firstly, analyzing the changes that could be affected by the dependency of artifacts, (e.g., classes or packages) on particular refactoring candidate - impact analysis. The changes might alter and create new artifacts for e.g., operations, classes or packages, which require a cost to do that. Therefore, the more dependencies the artifacts are, the more cost should be spent. For e.g., see `DI001` in Table III, `GodClass` depends on the other five classes and two external packages. Two points or weight will give to the fifteen classes and five points to 2 packages, (i.e., (2×15) + (5×2) = 40 points) as shown in *Refactoring Impact* column in Table III. Secondly, the defect likelihood could be analyzed by computing on how many defect fixes affected by the detected smells. The likelihood could be computed by detecting the smells, (e.g., specifically the god class) from certain periods, (e.g., from April to July). Then, count the number of defects that lead to fix in the god class in this time period and divide by the number of all defects that were fixed in the same time period. The higher the value the more likely a defect will be indicated in the god class. For instance, see column *Defect Likelihood* for `DI001`, it has been detected that the `GodClass` was god class from particular period. Assume the likelihood of 0.5, it means every second fixed defect will lead to changes in this god class. Thirdly, the change likelihood could be analyzed by computing on how likely a class is to be modified when a change to the software is executed. The same computation method will be used as defect likelihood for this purpose. It means the higher the value, the more likely that maintainability effort is higher for the god class [4]. For example, if change likelihood of 0.1 shows that the class was, on overage, modified with every 10^{th} change to the software. By computing the impact analysis, defect and change likelihood represent as a weight, it will, then, multiply by raw benefit. The raw benefit is an effort estimation that can be saved in terms of maintenance work in the next release. An expert will give this raw estimation. Then, the total benefit will be calculated. Based on the estimated cost and benefit values, the ROI value is calculated by (adopted from [8] where ***ROI = (Saving Effort and Less Impact of Proposed Refactoring/Effort of Proposed Refactoring))***, i.e., ratio of total Benefit to the total Cost. If the ROI value is greater than or equal to one, the refactoring is cost effective,

 ISBN: 978-1-61208-438-1

TABLE III. COST-BENEFIT ANALYSIS EXAMPLE

No.	Debt Item ID	Refactoring	Cost (Principal + Interest)	Risk (R) in Hour	Total Cost (Cost + R)	Refactoring Impact (RI)	Change Like. (CL)	Defect Like. (DL)	Weight (RI X CL X DL)	Raw Benefit (RB) in Hour	Total Benefit (Weight X EB)	ROI (TB / TC)	Rank
1.	DI001 -God Class -Cyclic Dependency	-Extract Class -Cut Dependency	22	-Regression bugs (2) -Testing (2)	26	40	0.5	0.1	2	5	10	0.4	1
2.	DI002 -Long Method Class -Inheritance too Deep	-Extract Method -Delegation	10	-Merge conflict (3) -Testing (2)	15	15	0.1	0.3	0.45	5	2.25	0.3	2
3.	DI003 -Duplicate Code -Cyclic Dependency	-Extract Method -Cut Dependency	8	-Build breaks(2) -Testing	12	10	0.25	0.11	0.28	3	0.84	0.06	3

i.e., the debt is paid off. Finally, the ROI values are prioritized. Based on the example (see Table III), `DI001` seems promising to be paid first instead of `DI002` and `DI003`. The ROI `DI001` value is bigger than the latter, (i.e., the refactoring effort could reach ROI) and it may give positive impact to the system.

D. TD Repayment

In the last phase, refactorings, which has been prioritized in the previous phase are added to the current backlog of the software system. By implementing the refactoring, the gap between the software as "it is" and the hypothesized "ideal" state could be closed. Although, there is no general agreement that refactoring could realize the idea, [9] claimed that by applying Test-Driven Development and continuous refactoring, the TD could be reduced systematically by releases. But, the questions "Which refactoring should be implemented first or later?" and "Should pay or not to pay?" are still open. Currently, we are still investigating how to strategically pay back based on TD metrics as introduced by [10].

IV. Related Works

Technical debt management. A few researchers have been focusing on how to manage TD. For example, [11] proposed a TD management framework, which aids managers to decide, which items should be implemented either first or later. A simple cost-benefit analysis is applied and less impact to the project is put at the top, i.e., prioritization. However, the approach does not consider risk factors in estimating the cost. Unlike CoBeTDM, it integrates risk factors [12] in the analysis due to uncertainty that may always happen. [13] introduced a tool to manage TD in terms of code violations. It guides to select the smells that should be refactored first based on pyramid data - the lowest part needs to be considered first. In contrast, CoBeTDM considers not only code but also architecture smells as the latter ones have high negative impact on the software quality.

Hotspot, code-, architecture-smells detection. We have adopted existing metrics [3] [5], which are quite useful to characterize smells on code- and architecture-level. However, these metrics do not integrate with each other. Our approach combines both metric sets to determine worst smells and identify very critical artifact as proposed by [14] for hotspot detection.

V. Conclusion

This paper introduces a TD management framework based on cost-benefit analysis, called CoBeTDM. It offers a systematic way of reducing the technical debt by quantifying cost and benefit of refactorings. It also considers with relevant risk factors. Until now, the CoBeTDM process is performed manually. But, we have started to develop a toolbox to support CoBeTDM and to monitor the TD trend in order to react early enough if the TD becomes critical.

References

[1] CAST, "Cast Worldwide Application Software Quality Study: Summary of Key Findings," 2010.

[2] C. Seaman, Y. Guo, N. Zazworka, F. Shull, C. Izurieta, Y. Cai, and A. Vetro, "Using technical debt data in decision making: Potential decision approaches," in 2012 Third Int. Workshop on Managing TD (MTD). IEEE, Jun. 2012, pp. 45–48.

[3] M. Lanza and R. Marinescu, OO Metrics in Practice - Using Software Metrics to Characterize, Evaluate, and Improve the Design of OO Systems. Springer, 2006, ISBN: 978-3-540-24429-5.

[4] N. Zazworka, C. Seaman, and F. Shull, "Prioritizing design debt investment opportunities," in Proceeding of the 2nd working on Managing technical debt - MTD '11. New York, New York, USA: ACM Press, May 2011, p. 39.

[5] M. Lippert and S. Roock, Refactoring in Large Software Projects: Performing Complex Restructurings Successfully. Wiley, 2006, ISBN: 978-0-470-85892-9.

[6] Hello2morrow, "Sonargraph Architect," 2013, URL: https://www.hello2morrow.com/products/sonargraph/architect [accessed: 2015-08-09].

[7] Sonarqube, "Technical Debt Calculation," March 09, 2011, URL: http://docs.sonarqube.org/display/PLUG/Technical+Debt+Calculation [accessed: 2015-09-08].

[8] R. Leitch and E. Stroulia, "Assessing the maintainability benefits of design restructuring using dependency analysis," in Proceedings. 5th Int. Workshop on Enterprise Networking and Computing in Healthcare Industry). IEEE Comput. Soc, 2003, pp. 309–322.

[9] J. Kerievsky, Refactoring to Patterns. Pearson Higher Education, 2004, ISBN: 0321213351.

[10] N. Ramasubbu, C. Kemerer, and C. Woodard, "Managing Technical Debt: Insights from Recent Empirical Evidence," IEEE Software, vol. 32, no. 2, Mar 2015, pp. 22–25.

[11] Y. Guo, R. O. Spínola, and C. Seaman, "Exploring the costs of technical debt management a case study," Empirical Software Engineering, Nov. 2014.

[12] M. Kim, T. Zimmermann, and N. Nagappan, "An empirical study of refactoring challenges and benefits at Microsoft," IEEE Transactions on Software Engineering, vol. 40, no. 7, 2014, pp. 633–649.

[13] J.-L. Letouzey and M. Ilkiewicz, "Managing TD with the SQALE Method," IEEE Software, vol. 29, no. 6, Nov. 2012, pp. 44–51.

[14] M. DAmbros, H. Gall, M. Lanza, and M. Pinzger, "Analysing Software Repositories to Understand Software Evolution," in Software Evolution SE - 3. Springer Berlin Heidelberg, 2008, pp. 37–67.

 ISBN: 978-1-61208-438-1

Design and Implementation of Business Logic Layer Object-Oriented Design versus Relational Design

Ali Alharthy
Faculty of Engineering and IT
University of Technology, Sydney
Sydney, Australia
Email: Ali.a.alharthy@student.uts.edu.au

Abstract—**Object-oriented programming has become one of the mainstream programming paradigms in software engineering, whereas relational models are predominant in commercial data processing applications. There is strong competition between these models for dominance in the building of modern applications, especially after the emergence and spread of object-relational mapping technology. This paper addresses the question of whether the object-oriented approach is better than the traditional approach in terms of flexibility with respect to changing requirements.**

Keywords-object-oriented design; relational design; requirement changes; maintenance

I. INTRODUCTION

Currently, most business logic layers of modern applications are constructed using either an object-oriented model or a relational model. The object-oriented model is based on software engineering principles such as coupling, inheritance, cohesion, and encapsulation, whereas the relational model is based on predicate logic and set theory principles [1]. The object-oriented model chains the building of applications within objects that have both data and behavior. The relational model supports the storage of data in tables and the treatment of that data with data manipulation language within the database through stored procedures and externally through structured query language. The relational model is currently used in many database systems [1]. Object-oriented technology is also commonly used in database application development. The difference between the two technologies is called the object-relational impedance mismatch [2][3]. In particular, when objects need to be stored in a relational database, object-relational mapping (ORM) appears to play an important role in overcoming the problem of impedance mismatch. ORM is a new technology that allows applications to access relational data in an object-oriented manner [4][5]. With the widespread use of ORM technology, domain objects are built as objects, and the application logic manipulates these objects in a pure object-oriented manner. The critical issue that arises is whether such an object-oriented model for business logic layers is a good choice in general. Proponents of the object-oriented approach have tended to assume that an object-oriented business model will make the system easier to maintain, easier to extend, and easier to reuse.

The object-oriented approach has been advocated as a tool for improving developer productivity and software quality [6][7]. Moreover, it has been suggested that development using object-oriented programming enhances productivity by simplifying understandability, program design, and maintenance in comparison to traditional approaches [8].These studies have maintained that using the object-oriented approach would help reduce the maintenance cost of software. However, there are few complete experimental results that support the claim that there is an advantage in the maintainability of programs developed with the object-oriented approach over those developed with traditional approaches [7][9].

The objective of this paper is to extend this body of knowledge by critically examining this assumption and to carefully compare the applicability and flexibility of the object-oriented system to those of the relational system. The findings from this project will be significant for practical applications in which the business logic layer is implemented in an object-oriented fashion, which is a growing trend in enterprise computing.

The rest of the paper is organised as follows. Section II presents the motivation for the study. Section III outlines the investigation method. Sections IV, V, VI, and VII present the case studies, and Section VIII reports the experimental results. Section IX concludes the paper.

II. MOTIVATION

Today, changing requirements have become a fact of life for software developers. Many studies have shown that changes in software were one of the reasons why various projects failed. For example, a study by the Standish Group found that only 37% of information technology projects are considered successes and that 21% of projects are considered failures [10]. The remaining 42% are considered ‘challenged’—defined as late, over budget, or having failed to meet expectations. Requirement changes are the major cause of this phenomenon. Such changes can occur during the development and maintenance phase in order to accommodate user and business requirements. Therefore, there is a need to identify a flexible approach that can deal with requirement changes.

However, ORM is very popular and widely used. According to Russell [3], in order to access data stored in relational databases, most modern applications are built using ORM technology rather than the traditional approach.

 ISBN: 978-1-61208-438-1

It has also been argued that using ORM tools can help reduce project costs. Moreover, proponents of the object-oriented approach have tended to assume that an object-oriented business model will make the system easier to maintain, easier to extend, and easier to reuse. On the other hand, proponents of the traditional approach have argued that not all the world must be handled in objects. In addition, they have maintained that there is some native incompatibility between ORM code and databases. They also maintain that although object-oriented development promises to reduce maintenance effort, these promises are not based on reliable experimentation [11]. Indeed, there is a significant lack of research on whether the object-oriented approach is better than the traditional approach in terms of flexibility in the face of requirement changes.

III. INVESTIGATION METHOD

The investigation is performed using a number of case studies and by introducing a variety of requirement changes in order to evaluate how the two approaches cope with them. For the implementation, we used Java Database Connectivity (JDBC), a representative relational system, and Hibernate, a representative ORM framework, as well as MYSQL, a relational database. All of these are popular open-source products. In order to measure the overall implementation effort associated with JDBC and Hibernate due to new/changed requirements, we used the code size produced in the completion of a task—the code size was measured in lines of code and takes into account lines added, modified, and deleted—as well as the time required to complete a task. To measure the code size, we used a free tool to compare the source code files after each implementation. The case studies implementation has been done by a developer who has six years experience in Web and Database applications development.

IV. FIRST CASE STUDY

We chose a simple case study to make an initial comparison of the effort involved in implementing the two technological approaches and changing them in response to requirement changes.

A. Problem statement

A company requires a Car Park application to maintain information about employees and their parking permits. The car park has a number of parking spots, which are divided into three areas: A, B, and C. Employees who want a permit have to pay a fee on a quarterly basis, which will be automatically deducted from their salary. The purpose of the Car Park application is to help the car park manager process the employees' applications for parking permits. Each employee has an ID, a name, and a phone extension. Each permit has a permit number, the car's registration number, and the section where the car can be parked. An employee can have at most two permits. Employees may change their extension in the course of their employment. When employees get a new car and want to use it instead of the old one, they have to discontinue the current permit and apply for a new one.

B. Comparison of the findings of the initial construction of the two approaches

TABLE I. FINDINGS OF THE INITIAL CONSTRUCTION

Program	Files	SLOC	Total/ Lines	ET
Hibernate	CPSystem.java	141	251	4 h 30 min
	Permit.java	47		
	Employee.java	47		
	HibernateUtil.java	16		
	Employee.hbm.xml	17	49	
	Permit.hbm.xml	14		
	Hibernate.cfg.xml	18		
	Total	300		
JDBC	CPSystem.java	283	283	3 h

Table I summarises the findings of the initial construction of the Car Park system using the two approaches. The table shows that even though there are no significant differences between the two approaches with respect to the effort measured by size of source code, the Hibernate approach took more time than the JDBC approach. In fact, with Hibernate we had to deal with six files, whereas with JDBC we had to deal with only one file. Therefore, the Hibernate approach took about 4.3 h, compared to 3 h for JDBC.

C. Impact of requirement changes on the two approaches

Because requirements change frequently in practice, it is useful to see how different approaches cope with requirement changes. For the initial investigation regarding requirement changes, we made the following change: in the Terminate Permit use case, instead of deleting the permit (as we did before), we labelled the permit as terminated.

D. Comparison of findings after first requirement change

TABLE II. FIRST REQUIREMENT CHANGE

Program	Files	V1	V2	A	M	D	S	ET
Hibernate	CPS.java	141	149	10	4	2	16	40 min
	Permit.java	47	65	8	0	0	8	
	Emp.java	47	47	0	0	0	0	
	Emp.hbm.xml	17	17	0	0	0	0	
	Perm.hbm.xml	14	15	1	0	0	1	
	Hiber.cfg.xml	18	18	0	0	0	0	
	HiberUtil.java	16	16	0	0	0	0	
	Total	300	327	19	4	2	25	
JDBC	CPSy.java	283	283	0	3	0	3	10 min

V1 = before the change; V2 = after the change; A = add; M = modify; D = delete; S = summation of A,M, and D; ET = estimated time

 ISBN: 978-1-61208-438-1

As Table II shows, there are significant differences between the two approaches with respect to the implementation effort measured by the size of the source code. The implementation of the new requirement changes with Hibernate required a total of 25 lines of code, compared to only 3 lines of code using JDBC. In addition, the implementation of the new requirement changes with Hibernate took about 40 min, whereas it took only 3 min with JDBC. Indeed, it is evident that JDBC offered more flexibility with regard to both time and effort.

E. Further impact of requirement changes on the two approaches

For the second requirement change, suppose a company needs to distinguish between full-time and part-time employees. Part-time employees are paid an hourly rate, whereas full-time employees are assigned a salary.

TABLE III. SECOND REQUIREMENT CHANGE

Program	Files	V1	V2	A	M	D	S	ET
Hibernate	CPS.java	149	154	5	2	0	7	1 h
	Permit.java	65	65	0	0	0	0	
	Emp.java	47	47	0	1	0	1	
	PartTime.java	-	20	20	0	0	20	
	FullTime.java	-	20	20	0	0	20	
	Emp.hbm.xml	16	16	0	0	0	0	
	Perm.hbm.xml	17	24	7	0	0	7	
	Hiber.cfg.xml	15	15	0	0	0	0	
	HiberUtil.java	18	18	0	0	0	0	
	Total	327	379	52	3	0	55	
JDBC	CPSy.java	283	296	13	3	0	16	20 min

As Table III shows, the new requirements have had a greater impact on the program implemented through Hibernate, in terms of both the time and the effort required to implement these changes. The implementation of the new requirement changes with Hibernate required a total of 55 lines of code, in contrast to JDBC, which required only 16 lines. This difference represents a nearly 3:1 ratio in quantity of code. Although one of the key benefits of inheritance is minimising the amount of duplicate code in an application by sharing common code amongst several subclasses, the majority of new code is due to inheritance code. Moreover, the implementation with Hibernate took about 1 h, compared to only 20 m using JDBC. As a result, increasing the number of classes that need to be persisted automatically can lead to increased levels of effort and time.

V. SECOND CASE STUDY

We made the second case study more complicated than the first in order to produce more statistics with which to compare the two approaches. We also made changes that reflect the change in business policy, that is, allowing more than one kind of item to be stored at a shelf location. This change in policy required a change in the structure of the classes. It will provide more data with which to compare the two approaches.

F. Problem statement

A database is needed to maintain information about the items stored in various warehouses of a company. Design a relational database, which can store the information contained the following:

1. Each warehouse has a phone (not shown) to contact the staff at the warehouse.
2. Shelf locations are of two types: single access and double access.
3. The present policies require that each shelf location, at any time, can be used to store only one kind of item.

TABLE IV. FINDINGS OF THE INITIAL CONSTRUCTION

Program	Files	SLOC	Total/ Lines	ET
Hibernate	PartInWareHouse.java	141	300	4 h
	Part.java	30		
	Warehouse.java	31		
	ShelfLocation.java	45		
	ShelfLocationPK.java	37		
	HibernateUtil.java	16		
	Warehouse.hbm.xml	12	58	
	Part.hbm.xml	13		
	ShelfLocation.hbm.xml	15		
	Hibernate.cfg.xml	18		
	Total	358		
JDBC	PartInWareHouse.java	202		2.3 h

Table IV summarises the findings for implementing the Parts in Warehouses system with the two approaches. Hibernate required a total of 358 lines of code, in contrast to JDBC, which required 202 lines. In addition, Hibernate required about 4 h, whereas JDBC required 2.3 h. Hibernate clearly required more effort and time than JDBC.

G. Impact of requirement changes on the two approaches

The storage rules change to allow more than one kind of item to be stored at a shelf location. This entails that the cardinality relationship between the two entities Shelf Location and Items must be changed to one-to-many.

H. Comparison of the findings after first requirement change

As shown in Table V, the new requirements have had a greater impact on the program implemented through Hibernate, in terms of both the time and the effort required to implement these changes. The implementation of the new requirement changes with Hibernate required a total of 139 lines of code, in contrast to JDBC, which required only 38. This difference represents a nearly 4:1 ratio in quantity of code. Indeed, the source of increase in code quantity was due to the addition of an item class with its composite key, which

 ISBN: 978-1-61208-438-1

is not necessary in JDBC. Moreover, the implementation with Hibernate took about 1.30 h, compared to only 30 min with JDBC.

TABLE V. FIRST REQUIREMENT CHANGE

Program	Files	V1	V2	A	M	D	S	ET
Hibernate	WHouse.java	141	155	14	5	0	19	1.30 h
	Part.java	30	30	0	0	0	0	
	Whouse.java	31	31	0	0	0	0	
	SLoc.java	45	32	2	6	15	23	
	SLocPK.java	37	37	0	0	0	0	
	Item.java	-	29	29	0	0	29	
	ItemPK.java	-	37	37	0	0	37	
	HibUtil.java	16	16	0	0	0	0	
	Who.hbm.xml	12	12	0	0	0	0	
	Part.hbm.xml	13	19	6	0	0	6	
	SLo.hbm.xml	15	20	5	2	1	8	
	Item.hbm.xml	-	16	16	0	0	16	
	Hiber.cfg.xml	18	19	1	0	0	1	
	Total	358	453	110	13	16	139	
JDBC	WHouse.java	202	218	16	20	2	38	40 min

VI. THIRD CASE STUDY: ISSUE OF RELATIONAL REPRESENTATION/NAVIGATION

The representation of the relationship is a fundamental issue. In fact, the difference between hierarchy, network, relational, and object-oriented databases is the way in which the relationship is represented. Therefore, if we construct the application with JDBC, we will not experience the navigation problem, whereas the problem arises when the application is constructed with ORM. Thus, we have to decide how to represent the navigation objects.

I. Problem statement

A distribution company supplies various kinds of products to customers on a daily basis according to the standing orders placed by the customers. The company wants to set up a system to maintain information about the products that the company can supply, its customers, and the standing orders.

J. Comparison of the findings of the initial construction of the two approaches

Table VI summarises the findings for implementing the Standing Order system with the two approaches. Hibernate required a total of 268 lines of code, in contrast to JDBC, which required 141. In addition, Hibernate required about 3 h, whereas JDBC required 2 h. Thus, Hibernate required more effort and time than JDBC.

TABLE VI. FINDINGS OF THE INITIAL CONSTRUCTION

Program	Files	SLOC	Total/ Lines	ET
Hibernate	SOSystem.java	86	212	3 h
	Customer.java	22		
	Order.java	47		
	Product.java	41		
	HibernateUtil.java	16		
	Customer.hbm.xml	10	56	
	Order.hbm.xml	15		
	Product.hbm.xml	12		
	Hibernate.cfg.xml	19		
	Total	268		
JDBC	SOSystem.java	141	141	2 h

K. Impact of requirement changes on the two approaches

We changed the navigation rule between the objects from unidirectional to bidirectional association.

TABLE VII. FINDINGS OF THE INITIAL CONSTRUCTION

Program	File Name	V1	V2	A	M	D	S	ET
Hibernate	SOSys.java	86	86	0	0	0	0	30 min
	Cust.java	22	32	10	0	0	10	
	Order.java	47	47	0	0	0	0	
	Product.java	41	51	10	0	0	10	
	Htil.java	16	16	0	0	0	0	
	Cu.hbm.xml	10	14	4	0	0	4	
	Or.hbm.xml	15	15	0	0	0	0	
	Pr.hbm.xml	12	16	4	0	0	4	
	Hib.cfg.xml	19	19	0	0	0	0	
	Total	268	296	28	0	0	28	
JDBC	SOSys.java	141	141	0	0	0	0	0

As Table VII shows, the new requirements have had a greater impact on the program implemented through Hibernate, in terms of both the time and the effort required to implement these changes. The implementation of the new requirement changes with Hibernate required a total of 28 lines of code and 30 min, in contrast JDBC, which did not require any changes, because navigation is not an issue for it.

VII. FOURTH CASE STUDY

We made this case study even more complicated and realistic in order to produce much more statistical data with which to compare the two approaches. The case study also highlights the issue of relationship representation and illustrates that the object-oriented approach is more sensitive to the class model than the relational model.

 ISBN: 978-1-61208-438-1

L. Problem statement

Eastern Suburb Gymnastics (ESG) is a regional organisation that is responsible for running competitions between the gymnastics clubs in eastern suburbs of Melbourne. The competitions are organised into seasons. ESG needs a system to help organise and maintain the records of the competitions that take place in a single season. The system, in essence, needs to store information on the gymnasts, their clubs, the organisation of the competitions, and the competition results.

M. Comparison of the findings of the initial construction of the two approaches

TABLE VIII. FINDINGS OF THE INITIAL CONSTRUCTION

Program	File	SLOC	Total/ Lines	ET
Hibernate	GScoringSystem	237	810	6 h
	Club	44		
	Competition	24		
	CompetitionPk	35		
	Division	60		
	EventPk	46		
	Event	37		
	EventType	58		
	Gymnast	60		
	Judge	40		
	Meet	52		
	TeamPk	46		
	Team	33		
	Score	22		
	HibernateUtil	16		
	Club.hbm.xml	13	177	
	Competition.hbm.xml	12		
	Division.hbm.xml	15		
	Event.hbm.xml	21		
	EventType.hbm.xml	14		
	Gymnast.hbm.xml	15		
	Judge.hbm.xml	17		
	Meet.hbm.xml	14		
	Team.hbm.xml	14		
	Score.hbm.xml	16		
	Hibernate.cfg.xml	26		
	Total	987		
JDBC	GScoringSystem	259	259	3 h

Table VIII summarises the findings for implementing the Eastern Suburb Gymnastics system with the two approaches. Hibernate required a total of 987 lines of code, in contrast to JDBC, which required 259. In addition, Hibernate required about 6 h, whereas JDBC required 3 h. It is evident that Hibernate required more effort and time than JDBC. This difference represents a nearly 4:1 ratio in quantity of code. Indeed, the source of the increase in the code quantity was due to a plain old Java objects (POJO) and its mapping files.

N. Impact of requirement changes on the two approaches

Here, we investigated how sensitive the two approaches are to the choice of domains modelled.

TABLE IX. FINDINGS OF THE INITIAL CONSTRUCTION

Program	File Name	V1	V2	A	M	D	S	ET
Hibernate	GSSystem	237	274	37	0	0	37	1 h
	Club	44	44	0	0	0	0	
	Competition	24	24	0	0	0	0	
	CompetitionPk	35	35	0	0	0	0	
	Division	60	60	0	0	0	0	
	EventPk	46	46	0	0	0	0	
	Event	37	37	0	0	0	0	
	EventType	58	58	0	0	0	0	
	Gymnast	60	60	0	0	0	0	
	Judge	40	40	0	0	0	0	
	Meet	52	52	0	0	0	0	
	TeamPk	46	46	0	0	0	0	
	Team	33	33	0	0	0	0	
	TeamMember	-	55	55	0	0	55	
	Score	22	22	0	0	0	0	
	HibernateUtil	16	16	0	0	0	0	
	Club.hbm.xml	13	13	0	0	0	0	
	Comp.hbm.xml	12	12	0	0	0	0	
	Divis.hbm.xml	15	15	0	0	0	0	
	Event.hbm.xml	21	21	0	0	0	0	
	EType.hbm.xml	14	14	0	0	0	0	
	Gymt.hbm.xml	15	15	0	0	0	0	
	Judge.hbm.xml	17	17	0	0	0	0	
	Meet.hbm.xml	14	14	0	0	0	0	
	Team.hbm.xml	14	14	0	0	0	0	
	TMember..xml	-	14	14	0	0	14	
	Score.hbm.xml	16	16	0	0	0	0	
	Hibern.cfg.xml	26	27	1	0	0	1	
	Total	987	1093	106	0	0	106	
JDBC	GSSystem	259	291	32	0	0	32	25 min

Table IX shows that the implementation of the new requirement changes with Hibernate required a total of 106 lines of code, in contrast to JDBC, which required only 32. This difference represents a nearly 3:1 ratio in quantity of code. Moreover, the implementation with Hibernate took about 1 h, compared to only 25 min for JDBC.

VIII. RESULTS

The results of our critical comparison of the two paradigms in terms of flexibility, which was based on implementation findings, indicate that in the initial construction of the application, using ORM is much costlier than using JDBC. In other words, the level of effort and time required to implement the application is much higher with Hibernate than with JDBC. For instance, the initial construction of the ESG system with ORM required a total of 987 lines of code, in contrast to JDBC, which required 259. This difference represents a nearly 4:1 ratio in quantity

 ISBN: 978-1-61208-438-1

of code. In addition, ORM required about 6 h, whereas JDBC required only 3 h. Indeed, increasing the number of classes that need to be persisted automatically can lead to increased levels of effort and time.

Moreover, JDBC is more flexible in the face of requirement changes than is ORM. For example, for an object to be persisted to a database, Hibernate needs a mapping file for all the objects that are to be persisted as well as POJO, which is not required when using the JDBC approach. This means that if we would like to add an attribute to or delete an attribute from a class, we must modify the mapping file of that class to map or delete the attribute, and subsequently we must modify the class itself to add/delete that an attribute with its getter and setter methods. When using JDBC, in contrast, we do not need to undertake these steps. Furthermore, the object-oriented paradigm has an issue related to navigation between objects through association links, whereas navigation is not an issue for JDBC. In addition, determining the direction with UML is not an easy task, which can be considered one of the common mistakes in design decision. In addition, the object-oriented approach is more sensitive to the class model than the relational model. It is worth mentioning that the developer did not use auto-code generation during performing the initial construction implementation, and this could explain the remarkable difference in time between two approaches.

Although the current study has yielded some clear preliminary findings, its design is not without flaws. First, the case studies were small scale as a result of some restrictions, such as the time and effort required for implementation. A further limitation is that the implementation of all the case studies was performed by one developer, which may affect the generalisability of the study's findings to different developers.

IX. CONCLUSION

This paper addressed the question of whether the object-oriented approach is better than the traditional approach or vice versa in terms of applicability and flexibility to requirement changes. The experimental results show that the object-oriented approach required more time and effort as a result of mapping files. Moreover, the object-oriented approach has an issue of navigation between objects. However, our examination is only the beginning. We believe there is still a need for further research with real projects to yield reliable results. Our future work will focus on conducting more experiments on real projects to validate our results and to investigate flexibility of object-oriented approach to requirement changes.

REFERENCES

[1] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, vol. 13, 1970, pp. 377-387.

[2] B. Unger, L. Prechelt, and M. Philippsen, The Impact of Inheritance Depth on Maintenance Tasks: Detailed Description and Evaluation of Two Experiment Replications. Fak. für Informatik Univ., 1998.

[3] C. Russell, "Bridging the Object-relational Divide," *Queue*, vol. 6, 2008, pp. 18-28.

[4] M. I. Aguirre-Urreta and G. M. Marakas, "Comparing Conceptual Modeling Techniques: A Critical Review of the EER vs. OO Empirical Literature," ACM SIGMIS Database, vol. 39, 2008, pp. 9-32.

[5] F. Lodhi and M. A. Ghazali, "Design of a Simple and Effective Object-to-Relational Mapping Technique," in Proceedings of the 2007 ACM Symposium on Applied Computing, 2007, pp. 1445-1449.

[6] S. Sircar, S. P. Nerur, and R. Mahapatra, "Revolution or Evolution? A Comparison of Object-oriented and Structured Systems Development Methods," MIS Quarterly, 2001, pp. 457-471.

[7] G. A. Kiran, S. Haripriya, and P. Jalote, "Effect of object orientation on maintainability of software," in Software Maintenance, 1997. Proc. International Conference on, 1997, pp. 114-121.

[8] M. B. Rosson and S. R. Alpert, "The Cognitive Consequences of Object-oriented Design," Human-Computer Interaction, vol. 5, 1991, pp. 345-379.

[9] M. A. Eierman and M. T. Dishaw, "The Process of Software Maintenance: A Comparison of Object-oriented and Third-generation Development Languages," Journal of Software Maintenance and Evolution: Research and Practice, vol. 19, 2007, pp. 33-47.

[10] S. Group. (2011). The Standish Group International Inc.

[11] E. Arisholm and D. I. Sjoberg, "Evaluating the Effect of a Delegated Versus Centralized Control Style on the Maintainability of Object-oriented Software," IEEE Transactions on Software Engineering, vol. 30, 2004, pp. 521-534.

 ISBN: 978-1-61208-438-1

Pymoult : On-Line Updates for Python Programs

Sébastien Martinez and Fabien Dagnat
IRISA, Télécom Bretagne
Brest, France
Email: first.last@telecom-bretagne.eu

Jérémy Buisson
IRISA, Écoles de Saint-Cyr Coëtquidan
Guer, France
Email: jeremy.buisson@irisa.fr

***Abstract*—On-line updates have proved to be essential for critical long running applications that hardly can be stopped. Indeed, security patches or feature enhancements need to be applied frequently. Pymoult is a platform allowing on-line updates for Python programs. It provides many mechanisms from the literature for updating running programs without requiring them to be stopped, allowing update developers to combine and configure the mechanisms for each update. This paper presents the design of Pymoult and details the implementation of several mechanisms it provides. With the help of an example, this paper also presents how mechanisms can be combined and configured to design on-line updates with Pymoult.**

Keywords–On-line updates; Python; Software maintenance

I. Introduction

Today's world expects software systems to be available at every moment, whether the system provides critical services like airport traffic control or whether its downtime would cause user discomfort like an operating system forcing a reboot for updating. Updating running software systems becomes a critical issue as it requires the system to be restarted, causing downtime and loss of state as well as financiary losses [1]. Not applying updates or postponing them is dangerous, as updates are necessary to keep software safe from bugs and security breaches. Dynamic Software Updating (DSU) allows updates to be applied on running software without requiring it to be restarted, causing little service disruption and no loss of data. This goal is reached by using DSU mechanisms for modifying the control flow (redefining functions) and the data flow (converting the data to a new version) of a given program. The majority of DSU platforms gather a predetermined set of these mechanisms they use to apply each update.

A lot of platforms have been proposed [2], [3], defining several mechanisms. Each mechanism has different properties and constraints. A DSU platform selects the best suited mechanisms for the type of program it targets and the kind of updates it expects. For example, K42 [4] is an operating system embedding its own DSU system. It handles its updates by swapping modified components when all old threads running out of date code are terminated. These mechanisms are best suited to the design of K42, which has a component based architecture and runs short lived threads. Updates often consist in the modification of components and, because the threads are short-lived, waiting for old threads to terminate is an easy way to ensure that components are swapped when they are quiescent. But when applying an unforeseen kind of update, the fixed set of mechanisms provided by the DSU system might be inefficient or even it may be impossible. For example, K42 does not handle API changes very well because they need to apply changes across the components.

Pymoult is a DSU platform providing several DSU mechanisms for updating Python programs. Its approach is to let an update developer select and configure the DSU mechanisms best suited for its update. While it requires more work from update developers than automated DSU platforms, it ensures that every update can be applied with best suited mechanisms. This paper presents the design and implementation of Pymoult. Section II discusses the implementation of DSU mechanisms in Python and presents Pypy-dsu, our custom Pypy interpreter enhanced for DSU support. Section III details the design of Pymoult and discusses the implementation of some of the mechanisms it provides before presenting an example of dynamic update using Pymoult in section IV. Section V compares Pymoult to other DSU platforms and Section VI introduces future work before concluding this paper.

II. Python and on-line updates

While many DSU mechanisms can be implemented in Python, some of them are impossible to develop using the standard implementation. For that reason, Pymoult uses Pypy-dsu, a Python interpreter enhanced with DSU features.

A. DSU capabilities of bare Python

Python is a dynamically typed, interpreted, object-oriented language. It has natural indirection and allows dynamic manipulation of programs models. The flexibility of Python and its introspection features make it easy to implement DSU mechanisms. For example, object fields, class methods, variables and functions are treated the same way, they are manipulated directly through their name. This allows, for example, to easily redefine a function `foo` by calling `foo=foo_v2` since each call to `foo` resolves the function name.

Fields can be added or deleted from objects and classes, allowing easy modification of objects or classes. The type of an object is kept as a `__class__` field which refers to that type. By consequence, changing the type of an object corresponds to changing the class the `__class__` field refers to and adding or deleting fields to conform the object to its new type.

Thanks to the meta-object protocol embedded in Python, Pymoult can implement a lazy method for updating objects. In Python, attributes and methods of objects are accessed (for writing, reading or calling) using `__getattribute__` and `__setattr__` methods of their class. By default, these methods resolve to the implementation in the `object` class. By overriding these methods for a given class, we can run updating code on an object before accessing its fields. Objects can therefore be updated only when actually used (i.e., when one of their fields is accessed).

Python is also a uni-typed language, allowing variables to change type dynamically without requiring specific tools. The type checking uses duck-typing. For example, if `a.foo` is called, the type of object `a` is checked for a method called `foo`. Variables can therefore be modified freely except for the deletion of fields used in the program.

 ISBN: 978-1-61208-438-1

B. The Pypy-dsu interpreter

Several DSU mechanisms can not be properly implemented in Python. For example, in a standard Python interpreter, it is not possible for a thread to suspend another one. This inability is a problem when needing to suspend parts of a program. We therefore decided to extend the features of a Python interpreter. We chose to base ourselves on the Pypy interpreter, a Python interpreter written in Python. Pypy is easier to modify than CPython (the reference Python interpreter) and already extends Python with object proxies and continuations that were helpful when implementing DSU mechanisms. This subsection presents new features that where added in Pypy-dsu, our customized Pypy interpreter.

1) Traces for controlling threads: Suspending a thread is implemented using traces. Python traces are functions called after each statement. To suspend a thread, a trace waiting for an event to be triggered is inserted. On the next statement, the trace will block until the event is triggered, causing the thread to be suspended. In Pypy, a trace cannot be set for a given thread and traces only start on the next call to a function. In Pypy-dsu it is possible to set a trace for a given thread using `sys.settrace_for_thread`. When setting the trace, one can choose whether the trace should start immediately or on the next function call. This feature allowed the development of mechanism to suspend and resume thread and control their execution (see paragraph II-B3 for an example).

2) Intercepting object creation: Although Pypy provides a garbage collector, it cannot be used to get a reference on every object created since the starting of the program. Such feature is essential when implementing mechanisms to update the data of a program. We therefore added the possibility to setup a global hook with `set_instance_hook` that is called each time an object is created. We use that hook to maintain a pool of weak references to each object created by the program. Each time an object is created, a hook creating a weak reference to it and adding it to the pool is called. This pool is used each time a mechanism requires accessing all the data at a same time.

3) Dropping frames: It is not possible in Python to manipulate the stack of a thread, making it impossible to support on stack replacement of functions. We added new instructions to drop frames from the stack. Calling a `dropNframe value` statement will cause the N most recent frames to exit immediately, returning `value`. On stack replacement of a function by a new one is implemented using traces and the `drop2frames` function to force the two most recent frames to exit and return value. A trace calling the new function before using `drop2frames` is inserted in the target thread. When the thread enters a frame running the old function, the trace captures the local state of that frame and calls the new function, giving that state as an argument. The return value of the new function is then given as argument to `drop2frames`. The last two frames (i.e., the frame of the trace and the frame of the old function) are dropped and the return value of the new function is returned to caller of the old function.

III. PYMOULT

To our knowledge, Pymoult is the first DSU platform for Python programs. Its approach is to provide as many DSU mechanisms as possible through an API that allows their combination and configuration. Since the creation of Pymoult in 2012, we implemented over 30 DSU mechanisms. For that reason and for the features we previously detailed, we think that Python is a good language for writing DSU mechanisms, and testing platforms designs. The design of Pymoult is the result of incremental work. Since the first version of Pymoult [5] and throughout the experiments we conduced with it, the design evolved to its actual form we present in this section.

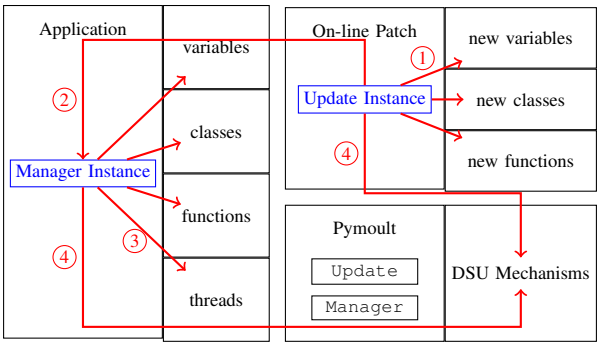


Figure 1. Map of an on-line update

To update a running program with Pymoult, the program developer must start a specific Pymoult thread called **`Listener`** in the program. That thread enables the supplying of on-line patches for the running application. An on-line patch is a piece of Python code that uses the Pymoult API. It contains the code of the updated elements of the program (e.g., functions, classes) and instructions on which DSU mechanisms to use. Dynamic updates rely on **`Manager`** and **`Update`** classes. A manager (an instance of the **`Manager`** class) is responsible for applying modifications according to the instructions given by an update object (an instance of the **`Update`** class).

Subsection III-A presents the design of Pymoult and details how to write an on-line patch with Pymoult. Subsection III-B details the implementation of some mechanisms provided by Pymoult and section IV presents the example of an on-line update using Pymoult.

A. Design

In Pymoult, an update is composed of several instances of an **`Update`** class. These instances are supplied to a manager that will apply them. For the remainder of this section, we use the term *update object* to refer to instances of **`Update`**. An *on-line patch* is therefore a set of update objects.

Figure 1 presents the architecture of a program undergoing an on-line update. An on-line patch embeds new variables, classes and functions ① which are used by an update object to specify the instructions for the manager ②. The manager controls and modifies the elements of the program ③ using DSU mechanisms provided by Pymoult and as specified by the update object ④.

Pymoult provides several off-the-shelf manager classes that can be instantiated in the program or in an on-line patch to create new managers. The regular **`Manager`** class describes a manager that operates only when the program calls its `apply_next_update` method. This kind of manager allows the program to decide when modifications can be applied to it. The **`ThreadedManager`** class describes a manager that operates in its own thread. It applies modifications each time an update object is supplied to it. Pymoult also provides preconfigured managers that are bound to an **`Update`** class and will always use the same DSU mechanisms to apply every modification. Lastly, one can extend the **`Manager`** class to

 ISBN: 978-1-61208-438-1

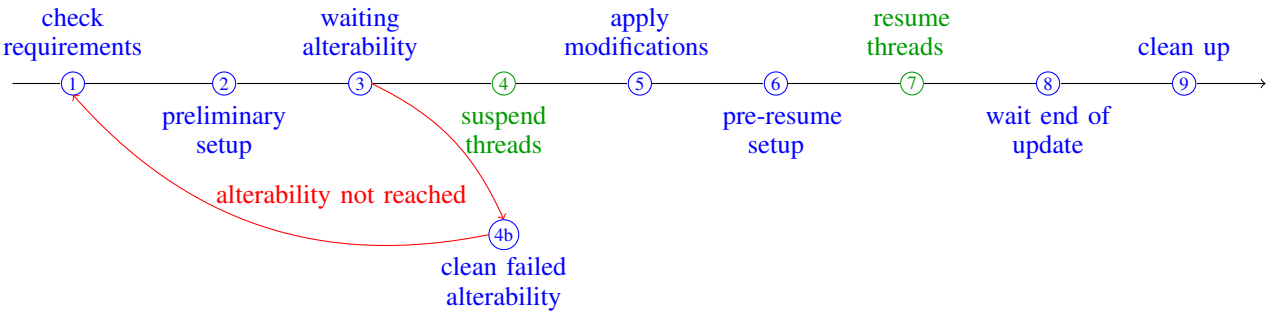


Figure 2. The updating process

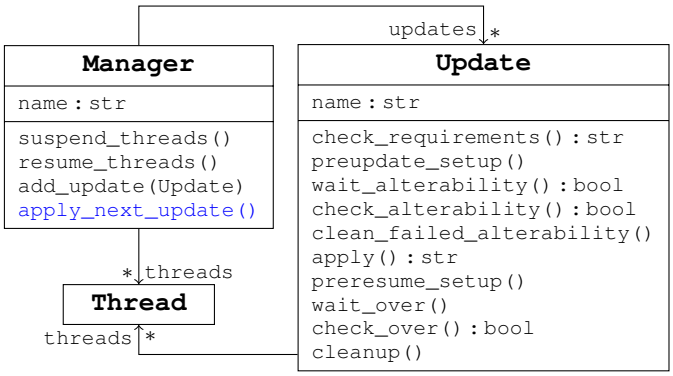


Figure 3. **`Update`** and **`Manager`** classes

define one's own manager. The classes involved in the updating process are presented in Figure 3.

Update developers can define their own update classes by extending the **`Update`** class. An update class has one method for each step of the updating process. These methods can use the DSU mechanisms provided by Pymoult through calls to specific functions. Update objects are instances of developer defined update classes and are supplied to managers. The managers implement the updating process pictured in Figure 2. When an update is supplied to a manager, that manager checks the requirements of the update ①. To do so, it calls the `check_requirements` method of the update that returns `"yes"`, `"no"` or `"never"` if the requirements are (respectively) met, not met or can never be met. If `"no"` is returned, the update is postponed. If `"never"` is returned, the update is canceled and if `"yes"` is returned, the updating process continues. The manager then proceeds to the preliminary setup step ② where it installs elements required for the next steps. To do so, it calls the `preupdate_setup` method of the update. When the preliminary setup is finished (i.e., the `preupdate_setup` method has returned), the manager waits for the application to be in a safe state we call *alterability* ③ by calling the `wait_alterability` method of the update. That method returns `True` when the application can be safely modified or `False` if a safe state could not be met in a fixed amount of time. If `False` is returned, the manager invokes a cleanup step ④b in which it calls the `clean_failed_alterability` method of the update for uninstalling the elements that were set up in the preliminary setup step. The update is then postponed. If `wait_alterability` returns `True`, the manager suspends some threads of the program ④ by calling its `suspend_threads` method. If the update specifies threads in its `threads` attribute, `suspend_threads` will suspend them, if not it will suspend the threads controlled by the manager (i.e., the threads in its `threads` attribute). If the manager does not control any threads, no thread is suspended. The manager then proceeds to the apply step ⑤ where it calls the `apply` method of the update. That method realizes all the modifications needed by the update (*e.g.* redefine functions, transform the data). The following step of the manager, the pre-resume setup step ⑥, calls the `preresume_setup` method of the update that follows the same principle as the `preupdate_setup` method. Suspended threads are then resumed by the `resume_threads` method of the manager ⑦. When all threads are resumed, the manager waits for the update to be over ⑧ by calling the `wait_over` method of the update that returns when the update is over. Indeed the apply step may have started tasks that run along the rest of the program. For example, the update can start lazy modifications of objects and requires all the objects to be transformed before completing. When the update is over, the manager cleans up any element installed in the preliminary setup and pre-resume steps ⑨ by calling the `cleanup` method of the update.

While this updating process is exactly followed as we just described by instances of **`ThreadedManager`**, instances of **`Manager`** wait passively for a safe state and for the end of the update. They give back the hand to the program each time they have to wait. For that purpose, they call the non-blocking `check_alterability` method (resp. `check_over`) instead of `wait_alterability` (resp. `wait_over`).

B. Mechanisms

Mechanisms are provided as functions that can be called in the methods of update classes. In this subsection, we follow the updating process detailed in the previous one and present some mechanisms that can be used for each step. Figure 4 presents an update object using the mechanisms discussed here.

1) Preliminary setup: Some mechanisms provided by Pymoult need preliminary installation before being used. This is the case of the `forceQuiescence` mechanism that forces a function to be quiescent. In the pre-update setup step, the `setupForceQuiescence` function replaces the targeted function by a stub that blocks all incoming, non-recursive calls by waiting for a specific *continue* event to be activated. A *watcher* thread is then started. That thread watches the quiescence of the targeted function.

2) Waiting alterability: We call *alterability* the state of a program when it can be updated without provoking errors. Indeed, if the update is applied at a wrong moment, updated code can call obsolete code and cause a crash. For example, an outdated function could try to access an updated piece of data that is no longer compatible with the function.

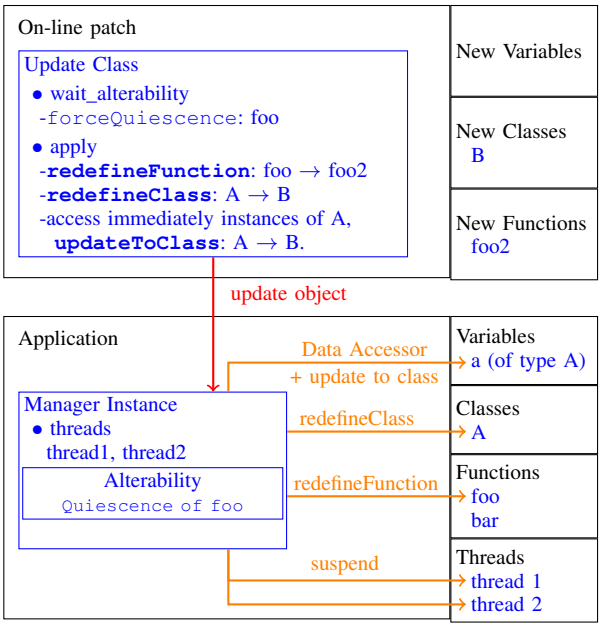


Figure 4. An example of update using Pymoult

Alterability can be detected by watching *alterability criteria* such as the *quiescence* of a component to be updated [6] or any condition on the state of the program. These criteria depend on the modifications applied by the update and may vary among all the updates. Several such criteria are proposed in the literature as the *tranquility* [7] or the *serenity* [8] of components. Pymoult provides several functions for expressing alterabilty crtieria. Here, we discuss the `waitForceQuiescence` function that expresses the criterion "target function must be quiescent" while forcing its quiescence instead of waiting for it. `waitForceQuiescence` waits for the *watcher* thread started in the previous step to detect the quiescence of the target function, then returns.

3) Applying modifications:

a) Accessing and updating data: Pymoult provides two ways to access data through the **`DataAccessor`** class that behaves as an iterator. When creating an instance of **`DataAccessor`**, one must precise the type of objects it accesses and the strategy to use as a string. The *immediate* strategy accesses all the objects when the instance of **`DataAccessor`** is created. It is then possible to iterate over all the objects. The *progressive* strategy uses the meta-object protocol described in section II-B to access objects lazily. Each time an object of the given type is used by the program, it is enqueued to the instance of **`DataAccessor`**. It is possible to iterate over the objects progressively as they are accessed. When the queue of accessed objects is empty, the iteration hangs until new objects are accessed. As a consequence, it is not possible to know a priori when all the objects have been accessed and therefore, when the iteration ends.

When they are accessed, objects can be updated using the **`updateToClass`** function. This function changes the type of a given object to a given class by updating its `__class__` attribute. A transformer supplied by the update developer is then applied to the object to modify its attributes.

b) Updating functions and classes: One way to update a function is to replace it by a new version. This mechanism is provided through the **`redefineFunction`** function that uses the native indirection of Python to change the body bound to the old function's name.

Similarly, classes can be redefined globally using **`redefineClass`** or one can add new fields or modify existing ones with **`addFieldToClass`**. In Python, classes are just special objects that can be modified dynamically as any other object.

4) Pre-resume setup: At this step, a mechanism may require some set up before resuming the excution of the program. For example, this is the case of `forceQuiescence`. In this step, the `cleanForceQuiescence` function activates the *continue* event waited by the blocking stub added during the pre-update setup. As a consequence, all the calls to the targeted function are released.

5) Cleaning failed alterabilty: DSU mechanisms handling alterability watching that required preliminary set up require clean up if the program fails to reach alterability. If the `forceQuiescence` mechanism fails to guide the program to alterability, the `cleanFailedForceQuiescence` function stops the *watcher* thread, activates the *continue* event and removes the stub installed in the pre-update setup step.

IV. An example

Various uses of Pymoult have been tested to validate it. Among the applications we have dynamically updated with Pymoult is the Django application server. Pymoult allowed us to update a running Django server from version 1.6.8 to version 1.6.10, choosing different DSU mechanisms for both successive updates (from 1.6.8 to 1.6.9 and from 1.6.9 to 1.6.10). Such a complex update does not fit as an introductory example. Instead, we present here the example of a program serving pictures through a socket. This program is representative of the Django example while staying simple.

Figure 5 presents the main elements of this program. Picture objects are stored in folders a `files` dictionary. Folders are served by the `serve_folder` method of the `ConnThread` class which defines connection handling threads. When starting, the program creates a listener to receive future on-line patches (as explained in section III). It also creates an instance of **`ThreadedManager`** and an `ObjectPool` that will contain weak references to all the created objects as explained in section II-B. That pool will enable immediate access to objects for future updates. Each time a new client connects to the server, a new `ConnThread` instance starts responding to all the commands it receives. The `do_command` method specifies the reaction to each received command.

Figure 6 presents an on-line patch that introduces support for comments. It is now possible to add a comment to pictures and before serving pictures from a folder, the pictures are annotated with their comment. The on-line patch redefines the `Picture` class and the `serve_folder` and `do_command` methods. In order to update the picture objects, the patch provides a transformer named `pic_trans`.

The `ServerUpdate` class defines a new update class which alterability criteria are the quiescence of the methods `do_command` and `serve_folder`. Because the update aims to redefine these two methods and to modify the picture objects they both use, waiting for their quiescence before updating ensures that it will not provoke errors. Before waiting

 ISBN: 978-1-61208-438-1

```
class Picture(object):
  def __init__(self,path,name):...
  def stream(self):...
class ConnThread(threading.Thread):
  def __init__(self,connection):...
  def serve_folder(self,folder):...
  def do_command(self,command):...
  def run(self):
    while self.connection:
      data = self.connection.recv(1024)
      self.do_command(data.strip())
def main():
  #create a socket to listen for commands
  while True:
    conn,addr = sock.accept()
    ConnThread(conn).start()
if __name__ == "__main__":
  listener = Listener()
  listener.start()
  manager = ThreadedManager()
  manager.start()
  ObjectsPool()
  main()
```

Figure 5. Structure of the program

for alterability, the update captures all the `ConnThread` instances and the main thread as they need to be suspended (Suspending the main thread ensures that no new `ConnThread` is created during the update). For that purpose, the patch defines the method `getAllConnThreads`. When alterability is met, the update uses **`addFieldToClass`** to redefine the methods and uses a **`DataAccessor`** to access the picture objects. It then uses **`updateToClass`** to update the accessed objects and **`redefineClass`** to redefine the `Picture` class.

The on-line patch creates an instance of `ServerUpdate` then supplies it to the manager. When the patch is sent to the listener created by the application, it is loaded in the application and its code is executed. The functions and classes it contains are defined and the update object is created and supplied to the manager.

Writing on-line patches as small programs which execution will update the targeted program allows for a fine control over the DSU mechanisms. For example, as presented in figure 7, we could have chosen to apply the update without waiting for the quiescence of `do_command` and `serve_folder` and use on-stack replacement to update theses methods while they are active. That would be a good choice if `do_command` and `serve_folder` are rarely quiescent at the same time. If the server handles a great amount of pictures, updating them all at the same time is long and disrupts the service since connections are suspended during the update. Updating picture objects lazily would be a better solution as data would be migrated without suspending connections (at the cost of the overhead introduced by the update of objects the first time they are accessed). Figure 7 presents this alternative patch for the update of the server. It uses **`rebootFunction`** to capture the state of currently running `do_command` and `serve_folder` methods then uses on-stack replacement. For that purpose, the patch defines the `command_capture` and `serve_capture` functions. The update uses **`startLazyUpdate`** to start updating picture objects lazily using the meta-object protocol described in II.

```
class Picture_V2(object):
  def __init__(self,path,name):
    ...
    self.commentary = "Witty comment"
    self.basepath = path
  def stream(self):...
  def comment(self,text):...
  def annotate(self):...
def getAllConnThreads():...
def pic_trans(pic):
  pic.basepath = pic.path
  pic.commentary = "Witty comment"
def serve_folder_v2(self,folder):...
def do_command_v2(self,command):...
class ServerUpdate(Update):
  def preupdate_setup(self):
    self.threads = getAllConnThreads()
  def wait_alterability(self):
    return waitQuiescenceOfFunctions([do_command,
                                      serve_folder])
  def apply(self):
    addFieldToClass(ConnThread,"do_command",
                    do_command_v2)
    addFieldToClass(ConnThread,"serve_folder",
                    serve_folder_v2)
    accessor = DataAccessor(Picture,"immediate")
    for picture in accessor:
      updateToClass(picture,Picture,Picture_V2,
                    pic_trans)
    redefineClass(Picture,Picture_V2)
conn_update = ServerUpdate(name="conn_update")
main.manager.add_update(conn_update)
```

Figure 6. Simplified on-line patch

```
class ServerUpdate(Update):
  def preupdate_setup(self):
    self.threads = getAllConnThreads()
  def wait_alterability(self):
    return True
  def apply(self):
    addFieldToClass(ConnThread,"do_command",
                    do_command_v2)
    addFieldToClass(ConnThread,"serve_folder",
                    serve_folder_v2)
    for thread in self.threads:
      rebootFunction(do_command,do_command_v2,
                     command_capture)
      rebootFunction(serve_folder,serve_folder_v2,
                     serve_capture)
    startLazyUpdate(Picture,PictureV2,pic_update)
    redefineClass(Picture,Picture_V2)
```

Figure 7. An alternate on-line patch (simplified)

V. Related work

To our knowledge, Pymoult is the only DSU platform for Python and its approach letting update developers combine and configure DSU mechanisms is an actual topic in the field. While classical DSU platforms use the same combination of mechanisms to apply every update, some platforms allow update developers to configure some mechanisms.

In ProteOS, [9] Giuffrida et al. propose to let update developers decide the alterability criteria for each update. The criteria are expressed as filters on the state of the OS. ProteOS allows processes to be updated by starting the new version of a process and transferring and updating the data from the old process to the new one. The data is accessed immediately using code instrumentation.

K42 [4] is an operating systems that allows its components to be swapped at runtime. When applying an on-line patch, it

 ISBN: 978-1-61208-438-1

forces all swapped components to be quiescent by suspending all threads created after the on-line update is requested and waiting for the old threads to terminate. Components are progressively swapped when they become quiescent.

Jvolve [10] is a DSU platform for Java programs. It allows classes and methods to be redefined. Update developers provide the source code of the program and of its updated version as well as class transformers (for updating static class fields) and object transformers (for updating object fields). The alterability criteria for every update is that the program must reach a VM safe point (usually a point where the garbage collector is called) where the redefined methods are quiescent. Update developers can also indicate methods whose quiescence will constitute an additional alterability criterion. When alterability is met, all threads are suspended and Jvolve updates methods using indirection at VM level and on-stack replacement. It accesses objects using the garbage collector and updates them immediately.

For these three platforms, the on-line patch supplied by the update developer is made of the source code of the new version of the program plus some instructions (K42: code of the transfomers; Jvolve and ProteOS: code of the transformers and of the alterability criteria). While Jvolve and ProteOS allow update developers to configure mechanisms by giving additional alterabilty criteria, they support little variability on the updating process. These platforms force the configuration of DSU mechanisms (*e.g* alterability criteria are quiescence of some functions) and only allow update developers to extend some of them (*e.g* by giving new functions that need to be quiescent for alterability). To our knowledge, Pymoult is the first DSU platform giving as much control on the mechanisms used for on-line updates.

VI. Conclusion and future work

We presented the design of Pymoult and presented how it allows DSU mechanisms to be combined and configured when writting an on-line patch. We also presented an example of on-line update of a Python program using Pymoult.

Pymoult is built atop a modified version of the Pypy interpreter. Because the modifications we applied to Pypy are little intrusive on the interpreter, they have no impact on the way Pypy interprets Python programs. Pymoult is therefore fully compatible with all the applications that are compatible with Pypy. Nevertheless, many common Python applications have compatibility issues with Pypy. The purpose of Pymoult was to find a design that allows DSU mechanisms to be easily configured and combined. Therefore, compatibility with every Python application was not an issue. Nonetheless, to ensure better compatibility with common Python software, we are developing a custom version of the CPython interpreter, the most used Python interpreter. Having a CPython-dsu interpreter will allow Pymoult to be tested with more real-life Python software.

Updating Django proved that Pymoult can be used to update real world software. Further experiments, such as overhead measurement, are required before validating the use of Pymoult for production software.

Pycots [11], a component model enabling architectural reconfiguration of applications, is an example of the use of Pymoult. The model is paired with a development process for specifying reconfiguration and proving their corectness using Coq before executing them using Pymoult.

The design of Pymoult is well suited for designing customized updates for Python programs. Having a similar design for different languages would be a good thing because it would allow combining DSU platforms for updating complex applications made of several programs using different languages. We are currently working on a C version of Pymoult as a means to establish an equivalent design for C programs.

Pymoult is free software published under GPL License. Its source code, as well as several examples can be found on the project repository [12]. The example presented in subsection IV is based on the "interactive" example.

Acknowledgement

The work presented in this paper is funded by Brittany regional council, as part of project IMAJD.

References

[1] Channelinsider, "Unplanned IT Outages Cost More than $5,000 per Minute: Report," http://www.channelinsider.com/c/a/Spotlight/Unplanned-IT-Outages-Cost-More-than-5000-per-Minute-Report-105393, 2011, [Online; accessed 28-September-2015].

[2] E. Miedes and F. D. Muñoz-Escoí, "A survey about dynamic software updating," Instituto Univ. Mixto Tecnológico de Informática, Universitat Politècnica de València, Tech. Rep. ITI-SIDI-2012/003, May 2012.

[3] H. Seifzadeh, H. Abolhassani, and M. S. Moshkenani, "A survey of dynamic software updating," Journal of Software: Evolution and Process, 2012. [Online]. Available: http://dx.doi.org/10.1002/smr.1556

[4] C. A. N. Soules et al., "System support for online reconfiguration," in Proc. of the Usenix Technical Conference, 2003, pp. 141–154.

[5] S. Martinez, F. Dagnat, and J. Buisson, "Prototyping DSU techniques using Python," in HotSWUp 2013 : 5th Workshop on Hot Topics in Software Upgrades, USENIX, Ed., 2013.

[6] J. Kramer and J. Magee, "The evolving philosophers problem: Dynamic change management," IEEE Trans. Softw. Eng., vol. 16, no. 11, Nov. 1990, pp. 1293–1306. [Online]. Available: http://dx.doi.org/10.1109/32.60317

[7] H. Chen, J. Yu, C. Hang, B. Zang, and P.-C. Yew, "Dynamic software updating using a relaxed consistency model," IEEE Transactions on Software Engineering, vol. 37, no. 5, 2011, pp. 679–694.

[8] M. Ghafari, P. Jamshidi, S. Shahbazi, and H. Haghighi, "An architectural approach to ensure globally consistent dynamic reconfiguration of component-based systems," in Proc of the 15th Symposium on Component Based Software Engineering, ser. CBSE. New York, USA: ACM, 2012, pp. 177–182. [Online]. Available: http://doi.acm.org/10.1145/2304736.2304765

[9] C. Giuffrida, A. Kuijsten, and A. S. Tanenbaum, "Safe and automatic live update for operating systems," in Proc of the International Conference on Architectural Support for Programming Languages and Operating Systems, ser. ASPLOS. New York, USA: ACM, 2013, pp. 279–292. [Online]. Available: http://doi.acm.org/10.1145/2451116.2451147

[10] S. Subramanian, M. Hicks, and K. S. McKinley, "Dynamic software updates: A vm-centric approach," in Proc of the Conference on Programming Language Design and Implementation, ser. PLDI. New York, USA: ACM, 2009, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/1542476.1542478

[11] J. Buisson, E. Calvacante, F. Dagnat, S. Martinez, and E. Leroux, "Coqcots & Pycots: non-stopping components for safe dynamic reconfiguration," in Proc of the 17th Symposium on Component-Based Software Engineering, ser. CBSE, ACM, Ed., New York, USA, 2014, pp. 85 – 90.

[12] S. Martinez, J. Buisson, F. Dagnat, A. Saric, D. Gilly, and A. Manoury, "Pymoult," https://bitbucket.org/smartinezgd/pymoult, 2008, [Online; accessed 28-September-2015].

 ISBN: 978-1-61208-438-1

Aiming towards Modernization: Visualization to Assist Structural Understanding of Oracle Forms Applications

Kelly Garcés, Edgar Sandoval,
Rubby Casallas, Camilo Álvarez
Los Andes University
School of Engineering
Department of Systems and Computing Engineering
Bogotá, Colombia
email:{kj.garces971,ed.sandoval1644,rcasalla,c.alvarez956}
@uniandes.edu.co

Alejandro Salamanca, Sandra Pinto, Fabian Melo
Asesoftware
Bogotá, Colombia
email:{asalaman, spinto, fmelo}@asesoftware.com

***Abstract*—Oracle Forms is a tool for creating screens that interact with an Oracle database. It appeared in the eighties and its use spread to many IT sectors today. There are pressures that push software engineers to modernize Oracle Forms applications: obsolescence of technology, requirements of users, etc. For a straightforward modernization, it is necessary to comprehend the applications from a prior step. This paper reports the preliminary results of the "Forms Modernization" project, in particular, of the understanding step. In most cases, the understanding of Forms applications is a complex and time-consuming task due to several reasons: large size of applications, lack of design documentation, lack of software organization. This paper proposes a visualization process to alleviate these issues. The process takes static Oracle Forms code as input and produces a set of domain specific diagrams/views, that ranges from high to low abstraction levels, as output. The gist of diagrams and views is to assist engineers in a structural understanding of the Oracle Forms software. The process includes algorithms for element discovery and clustering, and is instrumented by means of a tool running on Eclipse Modeling technologies. We take advantage of four real Oracle Forms applications to illustrate the benefits of this approach. These applications have been provided by Asesoftware, which is the Colombian industrial partner of the project.**

***Keywords*—program comprehension; reverse engineering; tools; clustering algorithms; model-driven engineering; graphical editors.**

I. INTRODUCTION

Software is constantly evolving; this evolution is motivated by different reasons such as the obsolescence of a technology, the pressure of users, or the need to build a single coherent information system when companies merge [1]. Our research lies in the field of software modernization, a kind of evolution, that refers to the understanding and evolving of existing software assets to maintain a large part of their business value [2].

This paper presents the preliminary results of the "Forms Modernization" project, which involves academic and industrial partners. The project arose as a result of some problems faced by Asesoftware, a Colombian software company that offers modernization services to its clients, regarding the desire to migrate Oracle Forms applications to modern platforms (in particular Java).

Oracle Forms appeared towards the end of the 1980s. It comprises a rapid database application development environment and a runtime environment, where the database applications run. Oracle Forms applications are present in many sectors. Such is the case in Colombia as well as in other countries. Results of a tool usage survey [3], carried out by the Oracle User Group Community Focused On Education (ODTUG) in 2009, indicate that 40 percent of 581 respondents (application developers) use Oracle Forms.

The migration of Oracle Forms applications to new technologies is mainly caused by three factors: the fear that Oracle desupports Forms, the difficulty to find Forms programmers, and Forms no longer meeting business requirements.

Furthermore, the company, Asesoftware, complains about the following three problems of manual modernization: i) Difficulty to understand the Oracle Forms application, ii) Time-consuming and repetitive migration, and iii) Poor testing. The "Forms Modernization" project addresses these problems in three phases. Here, we report the results of the phase that aims to solve the first problem.

According to Lethbridge and Anquetil [4], the main difficulties when trying to understand legacy applications are the following: i) lack of a directory hierarchy and of design information, ii) original designers' lack of knowledge of software architecture, and iii) undermining of the original design decisions as many additions and alterations were made. An Oracle Forms system is not the exception to Lethbridge and Anquetil's claim about legacy software organization: it lacks a directory hierarchy and the file names are not necessarily meaningful. As a result, an inspection of this code, aimed at understanding, is time consuming and error-prone.

To cope with this, Asesoftware organizes meetings with the clients, where the latter transfer their knowledge regarding application functionalities to the engineers in charge of the modernization process. The purpose of these meetings is to obtain a global understanding of the application's functional requirements, in order to ease the subsequent inspection of the code as well as the migration process. Nevertheless, this understanding remains in the mind of the engineers and it is not reported in any formal document in a way that the learning curve could be shortened for new people that enter the modernization process.

This paper presents the proposed approach as a solution to

 ISBN: 978-1-61208-438-1

the understanding issue. The approach consists of a process that takes a given Oracle Forms application as input and produces a set of diagrams and views that give an insight into the application's structural organization as output. The process includes algorithms for element discovery and clustering, and is instrumented by means of a tool running on Model-Driven engineering technologies. The resulting diagrams and views are designed to satisfy three concrete understanding challenges. When comparing our approach to related work —either research work [5][6][7] or commercial tools (i.e., Oracle2Java [8], Evo [9], Jheadstart [10], Pitss [11], Ormit [12])— we found that they only provide views with a low level of abstraction, whereas our approach proposes diagrams and views that range from high to low abstraction levels, thus, contributing to the acceleration of the understanding of the Oracle Forms program and aiming at modernization.

The paper is structured as follows: Section II describes the main building blocks of Oracle Forms applications and introduces four real Oracle Forms applications that serve as illustrating examples. Through an example, Section III elaborates on the understanding challenges that guide our research. Section IV establishes certain criteria, classifies related work according to it, and compares these works to our proposal. Sections V and VI present our approach and the tool used for instrumentation, respectively. Section VII describes how the user interacts with the visualizations in order to achieve an understanding. Section VIII elaborates on the results of applying our proposal to the illustrating examples. Finally, Section IX concludes the paper and outlines future work.

II. Oracle Forms Overview and Illustrating Examples

We present the main concepts of an Oracle Forms application below:

- Form: A Form is a collection of objects and code, including windows, items, triggers, etc.
- Blocks: Represent logical containers for grouping related items into a function unit to store, display and manipulate records of database tables. Programmers configure the blocks depending on the number of tables from which they want to manipulate the form:
  - The way to display a single database table in a form is to create a block. This results in a single table relationship between the form and the table.
  - The way to display two tables that share a master-detail relationship (i.e., "One to Many" relationship) is through two blocks. Oracle Forms guarantees that the detail block will display only records that are associated with the current record in the master block. This results in a master/detail relationship between the form and the two tables.
- Item: Items display information to users and enable them to interact with the application. Item objects include the following types: button, check box, display item, image, list item, radio group, text item and/or user area, among others.
- Trigger: A trigger object is associated to an event. It represents a named PL/SQL function or procedure that is written in a form, block or item. PL/SQL is the Oracle procedural extension of SQL. PL/SQL allows programmers to declare constants, variables, control program flows, SQL statements and APIs. A useful Oracle Forms API written in PL/SQL is the one offering procedures for form displaying, i.e., the OPEN/CALL statements.
- Menu: Is displayed as a collection of menu names appearing horizontally under the application window title. There is a drop-down list of commands under each menu name. Each command can represent a submenu or an action.

These concepts are found in the examples that will be used throughout the paper. These examples are aligned with four real applications related to treasury, banking and insurance sectors. These applications will be referred to as Conciso, Servibanca, Maestro, and Sitri. The following information is useful in order to give an idea about the application's size: the number of forms ranges from 83 to 178, referenced tables from 101 to 200, blocks from 361 to 765 and triggers from 2140 to 4406.

III. Challenges illustrated by an example

Using a concrete example, this section presents the challenges we face. Suppose a form of Conciso has to be modernized in two senses: i) evolution towards a new technology, and ii) introduction of a small modification to the initial functionality. The form allows manipulating deductions from an Employee's withholding tax. The modification consists in taking into account the deductions to which an employee has the right after making donations to institutions that promote culture, sports and art at a municipal level. Specifically, this modification should ensure that the user indicates a city, department and country in the form when the option of deduction by donations to local institutions has been chosen.

We face the following challenges as we try to understand the scope of the modernization at an application level:

A. Challenge 1: Functional modules and their relationships

This challenge concerns the following questions: *What is the functional module that contains the form subject to modernization? Is this module related to another modules?*

The fact of knowing the module that contains the form subject matter of modernization helps engineers to delimit the modernization scope. As we said in the introduction Section,the Oracle Forms software often lacks documentation, directory hierarchy and meaningful naming conventions; as a consequence, the functional modules are hard to infer. This is the case in the scenario where the client provides a folder that contains 144 forms on the root, with no subfolders nor documentation. Each form has a name that is the concatenation of a prefix (e.g., *CBF*) and a 5-digit number. In addition, the Oracle forms IDE only shows one form at a time, so that there is no a notion of a forms container.

Furthermore, it is important to know the dependency relationships between modules. A dependency relationship

between modules results when forms $a_1, a_2, .., a_n$ call forms $b_1, b_2, .., b_n$, and the forms are contained in two different modules A and B. Engineers can use this kind of relationship as an indication about the potential impact that a modification in the deduction form has on forms that belong to different modules. As for the modules, it is also hard to derive the relationships between modules in Oracle Forms. To do so, it is necessary to inspect the form PL/SQL code and look for CALL/OPEN statements directed towards another form. Note that these statements are spread along the form elements, which makes it difficult to discover the relationships between modules.

B. Challenge 2: Relationships between forms and tables

When addressing this challenge one should be able to answer the following questions: *Which are the tables related to the form that will be modernized? What is the type of relationship between the form and the tables?*

The relationships between forms and tables are important because they suggest to engineers that they have to review, more in detail, how changes in tables (for example, adding a foreign key from the deduction table towards the city table) impact form elements and their embedded PL/SQL code. The amount of effort to find out the tables related to a form depends on the type of relationship. Whereas single table and master/detail relationships are relatively easy to discover, by regarding the form navigation tree available on the Oracle Forms IDE, relationships resulting from references to tables embedded into the PL/SQL code are more time demanding because the code is scattered throughout the form elements (i.e. forms, blocks, items).

C. Challenge 3: Relationships between forms

This challenge includes providing an answer to the question: *Is there any form related to the form that will be modernized?* The purpose of this question is twofold: i) to know if related forms have to be changed in order to fully satisfy the functionality of the form after its modernization, and ii) to figure out if changes to the form subject matter of modernization impact the capabilities of the related forms. The example mentioned at the beginning of this section illustrates the first part of the purpose: it is important to know if there is any form —currently calling the deduction form— that needs to be modified in order to specify the different options of deductions and display the deduction form in an appropriate manner by taking into account the selection made by the user. This challenge is related to the first one in the sense that the relationship between two modules depends on the relationships between the forms that are contained in the modules. The mechanism to infer the relationships between forms is to review the PL/SQL code seeking for CALL/OPEN statements. Because this task has to be performed regardless of whether the forms are in the same module or in two different modules, it is very time consuming.

The challenges above are valid for multiple scenarios; they motivate the approach we propose in Section V. However, before elaborating on the approach, we present related work that helps us establish a background regarding visualization processes.

IV. Related Work

In this Section, we establish criteria that help us classify related research. For each criterion, we give a definition, variations on how the criterion can be satisfied —which results in categories—, and the position of each related work within these categories. The Section ends by comparing our approach to those found in related work.

A. Software systems

Tilley [13] has conducted extensive research into the use of views as a form of program comprehension. He found that numerous approaches focus on three different categories of software: i) legacy systems, ii) web applications, and iii) application design patterns. After considering the results of referential databases, we resolved that Tilley's criterion to classify view-related works is still valid, and decided to use it in our classification.

The legacy systems category encompasses traditional systems characterized as follows: monolithic, homogeneous, and typically written in third generation procedural programming languages. The purpose of related work within this category [5][6][7] is to achieve an understanding of the system, that can serve as a basis for its maintenance or for migration to newer languages.

The second category comprises Web applications. These systems often share many of the negative features of traditional legacy systems (e.g., poor structure, little or no documentation). The gist of related work in this category [14][15] is to understand the interaction behavior of the Web application, for further development and maintenance.

Finally, the third category covers a broader range of systems, including the software systems mentioned above. However, the difference is that related research within this category [16] specializes on design pattern recognition for better comprehension. The provided views are important to detect the critical points of an application for maintenance purposes.

B. Process

This criterion describes the steps that are followed to generate software systems views. Most of the reviewed approaches [5][6][7][14][16] agree with the following three steps: i) data injection, ii) querying, and iii) visualization. The first step consists in obtaining an in-memory representation from the input software artifacts. The second step aims at building blocks through the representation. Finally, the third step includes the generation of views for the groupings of blocks that result from the second step.

C. Input

This criterion indicates which kinds of inputs can be used by the process mentioned above. Literature reports mainly two kinds of inputs: i) Static input, and ii) Dynamic input. A static input only refers to source code [5][6][7]. Dynamic inputs, in contrast, are related to run-time information. Authors [14][15][16] obtain the second kind of input by executing scenarios that help them identify the invocation of a specific software feature (e.g., field, method, web page). Commonly, dynamic inputs are complemented by static inputs.

 ISBN: 978-1-61208-438-1

D. Source code language

This criterion points out the languages in which the source code is written; there are two categories: i) language specific, and ii) language independent. While the first category classifies the works whose implementations can be applied to source code written in a particular programming language, the second category encompasses the tools that can be applied to a variety of languages. Most of works [5][6][7][14][16] fall in the first category, they reverse engineer applications written in PHP, COBOL, Smalltak, C++, Oracle Forms. In contrast, there are few approaches in the second category [17]. The strategy of the latter is to develop bridges (i.e., programs, compilers, grammars, transformations) that allow the authors to go from the source code to an intermediate format on top of which the views are built up.

E. Notation

This criterion determines whether the notation used in the view is: i) standard, or ii) domain specific. The second option is suggested over the first one in cases where the reverse engineering task includes experts/users for which a customized graphical notation results in straightforward comprehension and communication. However, the second option implies a higher development effort when compared with the first one: while the first option can reuse existing viewers, the second option often requires the construction of viewers from scratch. The most disseminated standard notation among the articles [5][14][17][18] is the Unified Modeling Language (UML), in particular, class and sequence diagrams. Another popular standard notation is the graph theory, where nodes and edges are generic enough to represent any kind of software element and relationship between elements. An example of an articles that uses graph notation is [16]. In turn, the following are articles that propose domain specific notations: [6][7].

F. Views

In this criterion, we take advantage of Lowe's taxonomy [19] to classify the proposed views according to related work. Lowe et al. arrange the views in two categories: i) high level, and ii) low level. The high level category covers the views suitable to directly support program comprehension. Examples of such representations are class interaction graphs, lists of possible components/modules/subsystems, and architectural diagrams. On the other hand, low level views are much too complex to provide any understanding of any non-trivial program. Examples of low level representations are basic block graphs, single static assignment representations, call graphs, and control flow graphs. The following are some related works that provide high level views: [5][6][14][16][18]. In turn, [7][18] fall on the low level view category.

G. Comparison

Taking into account the criteria mentioned above, we compared our approach to related work —research work and commercial tools included— and we reached the following conclusions:

- *Software systems*: Similarly to [5][6][7], our approach falls on the legacy system category.

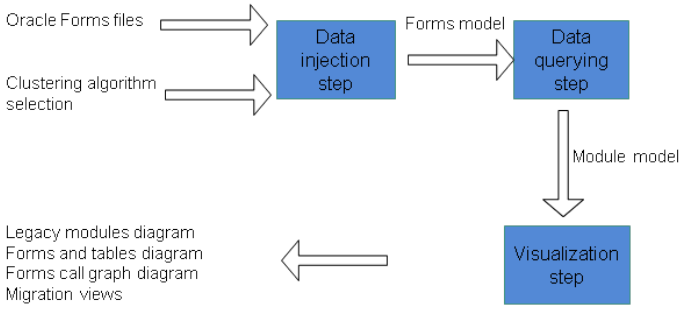


Figure 1. View-generation process overview

- *Process*: Our solution overlaps all previously cited solutions in the three steps of the view-generation process.
- *Input*: In similar fashion to [5][6][7], our approximation takes only source code as input.
- *Source code language*: Our approach takes source code written in Oracle Forms as input. That is, it belongs to the language specific category as well as [5][6][7].
- *Notation*: Like [6][7], our solution proposes a domain specific notation.
- *Views*: There is a noticeable difference between our approach and related work with respect to this criterion. Our literature review points out [7] as the sole scientific approach that provides views for Oracle Forms program understanding. The review also includes a set of commercial tools (i.e., Oracle2java, Evo, Jheadstart, Pitss, Ormit) that propose views for the same purpose. In both cases, the views are of two kinds: i) layout view and ii) application navigation tree. The layout view reflects how the graphical elements are arranged on a form and displayed to the user. The application navigation tree provides a hierarchical display of all forms in an application as well as the objects in each form —triggers, blocks, program units, etc.—. In our opinion, these views would be categorized as low level since they show a high level of detail; in contrast, we provide not only low level views but also high level ones, which can accelerate the understanding of the Oracle Forms program.

V. VIEW-GENERATION PROCESS

The view-generation process (see Figure 1) involves reverse engineering the Oracle Forms application and presenting several different diagrams and views to the developers. These diagrams and views can be further analyzed to determine subsystems, the elements that compose these subsystems(e.g., forms, database tables), and the relationships between these elements.

A. Data injection step

This step corresponds to data injection, which is the first step of a classical view-generation process (see Section IV-B).

 ISBN: 978-1-61208-438-1

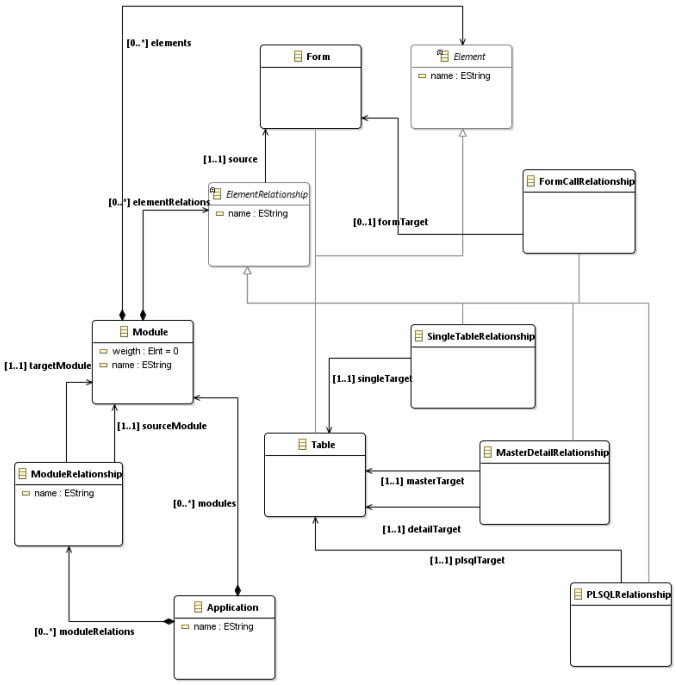


Figure 2. Module metamodel

As mentioned before, the purpose of this step is to obtain an in-memory representation from the input software artifacts. In our approach, we obtain an Abstract Semantic Graph (ASG) from Forms files (.fmb, .mmb). While a .fmb file describes a particular form, a .mmb file describes the menu from which all the application forms are displayed. This ASG is navigated to create model elements that conform to the Form metamodel. The main concepts of this metamodel have already been mentioned in Section II, namely forms, menus, blocks, items, triggers, relationships and tables. It is worth noting that we manage to extract not only the tables that are directly referenced by blocks, but also table references embedded into PL/SQL code. The reason to use models instead of the ASG is that we use tools that easily build editors for diagrams and views on top of models (see SectionVI).

B. Data querying step

This step corresponds to the second step of the view-generation process, whose gist is to search for building blocks through the representation. In our case, the representation is the Form model mentioned in the previous step and we search for elements such as modules, forms, tables, and relationships. Then, the elements resulting from this search are represented in another model, referred to as *Module model*. In contrast to the former model —which is verbose—, the latter model contains only the elements that matter in the visualization step. We describe the main concepts of the module model below and, then, the algorithms used to obtain it.

1) Module model: This model conforms to a metamodel (see Figure 2) and its concepts are explained below:

- *Application* is the root element of the metamodel. It describes the Oracle Forms application under study. An application consists of a set of modules that are related to each other.
- *Module* is a necessary concept because it works as a container of Oracle Forms elements and their relationships.
- *ModuleRelationship* represents the relationship between a pair of modules. A relationship going from module A to module B means that A contains a form that calls a form from B.
- *Element* describes Oracle Forms elements, i.e., forms and tables.
- *Form* specifies an Oracle form.
- *Table* indicates a table referenced from a form.
- *ElementRelationship* represents a relationship between a pair of Oracle Forms elements. An ElementRelationship can be classified into one of the following four sub-kinds.
- *SingleTableRelationship* is established between a form and a table, when the form has a block that references that table.
- *MasterDetailRelationship* is established between a form and two tables, when the form has two blocks —related via properties—, the tables are those referenced by the blocks, and one of them has the master role and the other one the detail role.
- *PLSQLRelationship* is established between a form and a table, if the form has PL/SQL that contains occurrences of the table name. A classic example of this is the relationship created from a form with a lookup field. The form contains a block with the addition of one field that displays data from another table. Such data is ”looked up” via PL/SQL code when the form runs.
- *FormCallRelationship* is established between two forms C and D, if form C contains CALL/OPEN statements parametrized with the name of form D.

2) Algorithms: Two kinds of algorithms are necessary to obtain a given module model:

1) *Element discovery algorithm*: This algorithm creates appropriate model elements depending on the forms, tables, and relationships found in the Forms model.
2) *Clustering algorithms*: One of the main concepts in the metamodel is the *Module*. Having modules makes the software easy to understand and, therefore, to change. However, it is not always easy to get the modules because legacy software organization is often quite poor. To cope with this, previous works in software comprehension[20][21] have used clustering algorithms. A clustering algorithm arranges software components into modules, by evaluating the relationships among these components. We have implemented the following two clustering algorithms that arrange the forms, tables and relationships discovered by the Element discovery algorithm, into modules:

 ISBN: 978-1-61208-438-1

a) Menu-based clustering algorithm: This algorithm takes a Forms model and its corresponding Module model —which results from the Element discovery algorithm— as inputs. From these two inputs, the menu-based clustering algorithm is in charge of producing a new Module model where the model elements (i.e., forms, tables, and relationships) are arranged into modules. For each menu in the Forms model, the algorithm inspects the commands in the respective drop-down list until it reaches the commands that are calls to forms. Then, the algorithm creates a module element —whose name is the corresponding menu name— and groups each form element within the module, according to the form name indicated by the corresponding call. In addition, the algorithm arranges the tables into the modules, following the relationships existing between forms and tables. Asesoftware Oracle Forms experts argue that there is good accuracy in the resulting modules diagrams when looking at the menus; however, they also point out that there is a lack of .mmb files because Oracle Forms programmers prefer to create menus by manually adding buttons through a .fmb file. We propose the following algorithm to tackle this lack of .mmb files.

b) Table betweenness clustering algorithm: This algorithm has four phases:

1) In the first phase, it takes a Module model —which results from the Element discovery algorithm— as input and produces a graph as output. In the graph, the nodes represent forms, and an edge is established between each pair of nodes (or forms) if they have several tables in common.
2) In the second phase, the algorithm determines the modules, that is, the subgraphs of the graph obtained in the first phase. This algorithm identifies a subgraph because its inner connections are dense, and the outer connections among subgraphs are sparser. There are several ways of identifying subgraphs, however, due to the delivery dates of the project being so close, we decided to use an existing method: the Girvan-Newman algorithm [22]. Therefore, in the second phase, our algorithm delegates the subgraph construction to the Girvan-Newman algorithm. The latter progressively finds and removes edges with the highest *betweenness*, until the graph breaks up into subgraphs. The betweenness of an edge is defined as the number of shortest paths between all pairs of nodes in the graph passing through that edge. If there is more than one shortest path between a pair of nodes, then each path is assigned equal weight such that the total weight of all of the paths is equal to unity; nonetheless, the betweenness value for an edge is not necessarily an integer. Because the edges that lie between subgraphs are expected to be those with the highest betweenness values, a good separation of the graph into subgraphs can be achieved by removing them recursively.
3) In the third phase, our algorithm creates a module element for each subgraph indicated in the Girvan-Newman algorithm output. For each node in a subgraph, the algorithm groups into the module the corresponding form element. To do so, the algorithm follows two rules: i) If a subgraph has more than one node (i.e., a form), the algorithm arranges the forms (and referenced tables) within a new module —whose name is the concatenation of a keyword and a counter—; ii) If a subgraph has only one node, then it is arranged into the *isolated form module*.
4) Finally, in the fourth phase, the algorithm arranges the tables into the modules, by following the existing relationships between forms and tables.

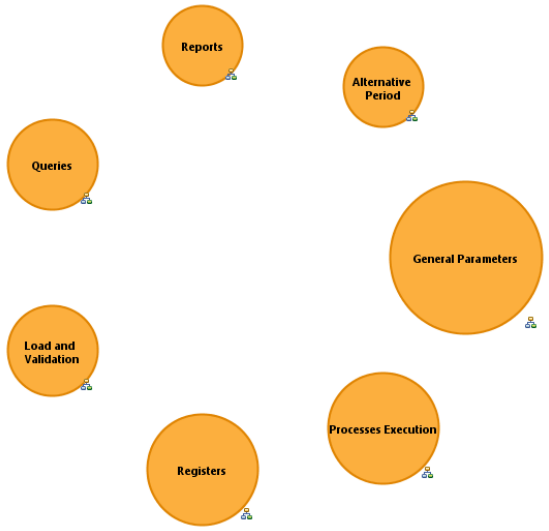


Figure 3. Legacy modules diagram for Conciso (result of the Menu-Based clustering algorithm)

It is worth noting that the number of database tables in common and the number of iterations of the Girvan-Newman algorithm, that is, the parameters used in the first and second phases, respectively, are given by the user and impact the number of resulting modules as follows: A highest number of database tables in common or a highest number of iterations result in the following: i) a highest number of modules, which are small in size because each of them contains few forms, and ii) an big-sized isolated form module that contains a lot of forms.

C. Visualization step

This step involves techniques that are of use to present the gathered information via diagrams and views. These diagrams and views are high-level or low-level representations that allow developers to obtain a structural understanding of the system. Basically, the diagrams have nodes and edges, and the views look like tables. We describe the different aspects of diagrams below: category (either low or high), purpose, notation, layout and filters that ease their navigation. The Section ends by presenting the information displayed in the table-like views.

1) Functional modules and their relationships: This diagram belongs to the high-level category. Its main purpose is to show how a legacy system is organized in terms of modules or subsystems, and which are the relationships between the modules of a system. A secondary purpose of the legacy module diagram is to serve as an entry point for the *Forms and tables diagram*. Figure 3 shows the diagram for Conciso, after applying the menu-based clustering algorithm (the notation is

 ISBN: 978-1-61208-438-1

the same as in the table betweenness algorithm). The notation used in both diagrams is explained below:

- An orange circle represents a Module. The circle label is the module name, which can be changed by the user into a more meaningful name. The size of the circle is proportional to the number of form elements contained within the module.
- A red arrow represents a ModuleRelationship.

The modules are radially arranged in descending order by size. The module with the biggest size is positioned at three-o'clock and the remaining modules are organized proceeding clockwise. In addition, the diagram provides a filter that hides ModuleRelationships.

2) Forms and tables diagram: This diagram falls into the low-level category. It is available when one selects a module from the legacy modules diagram. Its purpose is to show the forms and tables contained in the module and the relationships between them. Figure 4 shows excerpts from the Forms and tables diagram of a module of the illustrating example (i.e., *General Parameters*). The notation that was used is explained below:

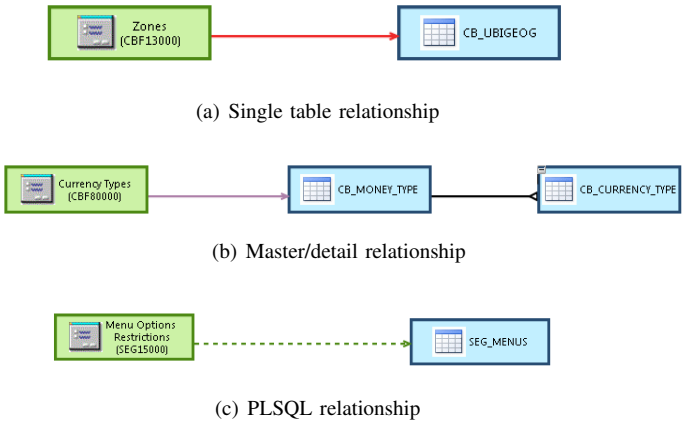
- A green square represents a Form. The square label is the form name (if present) or the file name that corresponds to the form.
- A blue square depicts a Table. The label is the table name.
- A red arrow indicates a SingleTableRelationship (see Figure 4(a)).
- A pair of purple and black arrows indicates a MasterDetailRelationship. In particular, the purple arrow points to the master table and the black arrow to the detail table (see Figure 4(b)).
- A green dotted arrow represents a PLSQLRelationship (see Figure 4(c)).

The diagram layout is in charge of placing all the elements in a way that the relationships intercept as little as possible. Furthermore, the diagram offers filters that allow us to leave all the relationships of a certain type visible in the diagram.

3) Forms call dependency diagram: This diagram belongs to the low-level category. This diagram presents the call-graph of the forms of an Oracle Forms application. Figure 5 shows an excerpt from the Forms call dependency diagram of the *General Parameters* module. The notation that was used is explained below:

- A green circle represents a Form. The circle label is a concatenation of the form name (if present) and the file name that corresponds to the form.
- The arrows describe FormCallRelationships between forms. In particular, a green arrow indicates an OPEN statement and a red arrow a CALL statement.

The forms are arranged following a tree layout. In addition, there are two kinds of filters: i) A filter that removes all unconnected Forms from the diagram, and ii) A filter that, if disabled, hides all FormCallRelationships.



(a) Single table relationship

(b) Master/detail relationship

(c) PLSQL relationship

Figure 4. Excerpt of Forms and tables diagram for Conciso.

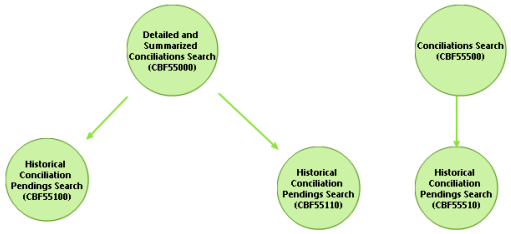
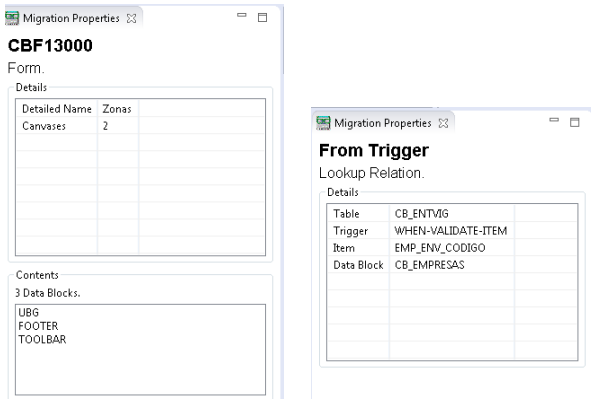


Figure 5. Forms call dependency diagram for Conciso.

4) Migration views: This view falls into the low-level category. It displays detailed information about an element, when it is selected by the user from one of the aforementioned diagrams.

- The *Module migration view* is displayed when a module is selected from the legacy module diagram. It shows the module's weight and the forms and tables it contains. Due to page restrictions, a figure illustrating this view is not included. It is worth noting that this view looks like the views below, but it displays different information.
- The *Form migration view* is shown when a form is chosen from the forms and tables diagram. It demonstrates the detailed form name, the number of canvases, and the blocks and program units declared in the form (see Figure 6(a)).
- The *Relationship migration view* is offered when a relationship is selected from the forms and tables diagram. The view shows the relationship details according to its type:
  - In case of a MasterDetailRelationship, it points out the master and detail tables, the Oracle Form relationship, and the block.
  - In case of a SingleTableRelationship, it shows the table and the corresponding block.
  - In case of a PLSQLRelationship, it shows the table, the respective block, and the trigger where the PL/SQL is embedded (see Figure

 ISBN: 978-1-61208-438-1



(a) Form migration view (b) PLSQL relationship migration view

Figure 6. Migration view examples for Conciso.

6(b) that contains the most complex migration view for relationships).

VI. TOOLING

We have built a tool that instruments our approach. The components that comprise the tool architecture are described below. Components 1 to 4 are part of an existing open source infrastructure[23] that we used to build the tool, but components 5 to 10 are built by us.

1) *Eclipse*: includes a basic platform (i.e. workbench, workspace and team facilities) that is useful for the development of extensions.
2) *EMF*: is a framework for building tools based on a metamodel.
3) *Acceleo*: includes a feature that interprets the Object Constraint Language (OCL). OCL is a language that provides query expressions on any model.
4) *Sirius*: is a plug-in to create graphical editors that allow edition and visualization of models. Sirius can be classified into the approaches provided with DSL constructs that serve to specify the graphical notation of a view, e.g., rules to determine color/size of nodes or edges, layout, etc. Sirius is applicable to any domain, for example, a Sirius node can represent a software system entity but also the member of a family.
5) *Domain metamodels*: contains the Forms and Module metamodels presented in Section V.
6) *Forms injector*: its purpose is twofold: i) to obtain a form model from Forms files, and ii) to enrich the Module model with PL/SQL relationships. In order to meet the first purpose, we take advantage of the JDAPI [24], which is an API to manipulate Forms files. Thus, we navigate the ASG —resulting from the JDAPI— and create model elements according to the Forms metamodel classes. To attain the second purpose, we adapt an existing PL/SQL ANTLR parser.
7) *Clustering algorithms*: these algorithms have been implemented as Java programs. In particular, the table betweenness clustering algorithm takes advantages of the betweenness centrality algorithm that comes with the JUNG API [25].
8) *Diagrams and views specification*: A model that, conforming to Sirius constructs, specifies the graphical notation of diagrams and views.
9) *Customized layout*: implements layouts beyond Sirius' default layouts (i.e., tree or composite). Currently, it contains the radial layout implementation, useful in the legacy modules diagram.
10) *Wizard*: is a graphical interface that allows engineers to configure visualization process aspects, that is, the Form files path, the form to be processed and the clustering algorithm.

VII. INTERACTION PATH

In this Section, we describe how the engineer, guided by a set of questions —stated in the form of challenges— uses the visualizations in order to achieve a progressive understanding of the Oracle Application. Like in previous sections, Conciso is used for illustration purposes.

A. Challenge 1: Functional modules and their relationships

The legacy modules diagram targets this challenge. Figure 3 shows the resulting diagram for Conciso. It contains seven functional modules and zero relationships between modules. We present below, how this diagram is obtained and how engineers can take advantage of it and of adjacent tooling to address Challenge 1. Given that Conciso includes .mmb files, we selected the menu-based clustering algorithm option first, from the visualization process wizard.

Once the process is finished, the view is derived; then, engineers have two options to figure out which module contains the deduction form —whose physical name is *CBF55410*— . On the one hand, the first option consists of the following two steps: i) To point each module displayed in the legacy module diagram and ii) To look at the Migration properties view of each module, until finding the deduction form in the *Contents* list. On the other hand, the second option includes the following three steps: i) To open Acceleo Interpreter, ii) To point the root of the Module model and iii) To build an OCL query to ask for the module tat contains the form.

Knowing that the module *General Parameters* contains the deduction form, engineers go back to the diagram to see the module properties: name, size, relationships, etc. The fact that the module *General Parameters* has no ingoing/outgoing relationships, is a signal to engineers that they only have to take care of propagating changes inside the mentioned module (if ever needed). There is no need to worry about other modules when modifying the deduction form. In addition, the diagram shows that there are no relationships between modules, which indicates that Conciso modules are decoupled enough.

We decided to derive another legacy modules diagram for Conciso. This time, we chose the table betweenness clustering algorithm from the visualization process wizard. When comparing the result of this algorithm with the menu-based clustering algorithm result, we observed no correspondences with respect to the number of modules and their content. This fact should not call into question the accuracy of the

 ISBN: 978-1-61208-438-1

algorithm. Instead, the reason for this disparity is that the algorithm's derive modules take into account different aspects of software. On the one hand, the menu-based algorithm uses the menu whose items are normally organized in terms of user tasks. As a consequence, the resulting modules diagram maps the final user mental model. On the other hand, the table betweenness algorithm organizes modules taking into account the tables common to different forms. Thus, the resulting modules diagram maps an internal view of the software that is potentially useful to developers. We conclude that in case of having .mmb files, engineers can use the two clustering algorithms results as complementary perspectives. However, in case of lacking .mmb files, the table betweenness is a good starting point for structural understanding.

B. Challenge 2: Relationships between forms and tables

Now, we describe how forms and the table diagram address this challenge. When located in the *General Parameters* module, engineers can navigate the forms and tables diagram until finding the deduction form. Engineers can focus the form following two alternative paths: manual scrolling or an Acceleo query. At the beginning, the diagram shows the relationships between all the forms and tables in the module (i.e. 19 single table, 37 master/detail and 407 PL/SQL relationships), which makes it difficult to focus on the elements that matter. Here is where the filters gain prominence: it is suggested that engineers firstly switch off all filters and, then, progressively turn on each one of them, going from the simplest (i.e., the single table filter) to the more complex (i.e., the PL/SQL filter). Every time a new filter is activated, engineers should analyze the resulting relationships. As an output of filtering, engineers conclude that the deduction form has only a master/detail relationship with two tables, where the master is *CP_MONEY_TYPE* and the detail *CP_CURRENCY_TYPE* (see Figure 4(b)). From this, they obtain knowledge about the tables whose change may impact the deduction form in any way. Subsequently, engineers should complement their knowledge with database information, in order to get a more precise insight about the nature of the impact. A benefit of the diagram, when compared with the manual approach, is that it points out only the tables that are relevant to the form —which would likely speed up the impact study.

C. Challenge 3: Relationships between forms

This paragraph describes how the Forms call dependency diagram targets the third challenge. Once the diagram is generated, engineers can observe that several forms have no dependencies with others. At this point, it is recommended to apply the filter *Single Elements* to leave only the forms that share dependencies. After filtering, engineers obtain the diagram shown in Figure 5. Given the call relationship between form *CBF55400* and the deduction form, engineers can infer that changes in the former will likely impact the latter. Then, engineers need to complement their knowledge with an external source (e.g., the Oracle Forms navigation tree) in order to specify the kind of impact on the related form. Like the Forms and table diagram, the forms call diagram benefit —when compared to a manual approximation— is that it limits the number of forms that have to be inspected during a subsequent impact review.

VIII. Applying approach to illustrating examples

To demonstrate the applicability of our approach, we have obtained visualizations, not only for Conciso but for all the applications mentioned in Section II. Below, we present a table that summarizes what we noticed regarding the resulting diagrams for these applications. Ultimately, we analyze the table data by taking the challenges into account. All tests were executed on a machine with a Windows 7 operative system, Intel Xeon dual core processor and 12 GB of RAM.

TABLE I. Visualization statistics for all applications

Criteria		**Conciso**	**Maestro**	**Servibanca**	**Sitri**
Clustering Algorithm		Menu -based	Table betweenness	Table betweenness	Menu -based
Modules		7	7	10	69
Module relationships		0	6	5	1
Forms		144	155	83	178
Forms and tables relationships	**Master**	87	52	42	95
	Master Detail	47	43	23	61
	PL/SQL	958	1234	462	1832
Forms relationships		3	154	30	1
Processing time (seconds)		62	83	49	90

- *Modules and relationships between modules:* The module row shows the application size in terms of modules; it ranges from 7 to 69. The latter number indicates not only that Sitri is the largest one from a functional perspective, but also that its menu may be complex due to the large number of options (i.e., 69). In turn, the modules relationships row demonstrates that the modules of Conciso and Sitri have a low coupling. Given that the menu-based clustering was used to derive the "legacy modules diagrams" for Conciso and Sitri, we conclude —from the number of module relationships— that each menu option calls forms that are not called from another menu option. In addition, the modules relationships row shows that Maestro and Servibanca have the highest coupling when compared with the rest of the applications. The rationale behind this result is related to the table betweenness algorithm parameters (i.e., number of database tables in common and number of iterations). It is worth emphasizing that the relationships between modules summarize the relationships between forms contained in different modules. This is the reason why the number of the former relationships is less than the number of the latter relationships.
- *Forms and relationships between forms and tables:* There is a correlation between these rows and the processing time row. The processing time results show that the time spent on the visualization process ranges from 90-50 seconds. The value for each application depends on the application size: the more forms and relationships (either single table, master/detail or PL/SQL), the longer the processing time. For example, Sitri, with the highest processing time, contains much more forms and relationships (i.e., 178 and 1988, respectively) than the rest of the applications.

 ISBN: 978-1-61208-438-1

- *Relationships between forms:* As shown in this row, Conciso and Sitri have few relationships between forms (i.e., from 1 to 3); this occurs because, in these applications, most forms work as independent units accessed through a menu. In contrast, Maestro and Servibanca have much more relationships (i.e., from 30 to 154); the reason is that these applications have no .mmb files. Instead, menus are created manually by adding buttons in .fmb files. As a result, many forms are dependent on these .fmb files.

IX. Conclusion and future work

This article proposes a visualization approach for Oracle Forms applications. The diagrams and views have been designed having the ease of modernization in mind. This approach has two main benefits, which were discussed in Section VIII and can be summarized as follows: i) The proposed visualization aids engineers to obtain an understanding of the application. This knowledge can be useful to determine the modernization scope at different abstraction levels: At a high abstraction level, it shows modules that could be potentially impacted by a change made to a form. At a low abstraction level, it points out forms and database tables that are likely affected by the change. ii) The second benefit concerns the productivity of engineers: when compared with the manual inspection of Oracle Forms assets, our visualization approach should reduce the understanding effort in terms of time. This claim will be formally validated by means of a focus group, before project closure.

Also, we outline the four fronts on which we are working on below: i) As was mentioned in Section VIII, the proposed visualization gives an initial knowledge that has to be complemented with information coming from other sources. The navigation from our tool to these sources and vice versa can be tedious, therefore, we are currently working on the integration of the most common source —the Oracle Forms IDE— into our tool; ii) The possibility to reorganize the modules from the diagrams in a way that the new organization is maintained during the migration process; iii) A new functionality that allows engineers to add information to the diagrams —this information could summarize the knowledge they have acquired from the visualizations and from external sources, such as final users—; iv) Finally, a new visualization that looks like a table to display application statistics —such as number of forms, blocks, trigger, etc.— would be desirable. These statistics can be helpful for engineers to estimate modernization costs.

References

[1] F. Fleurey, E. Breton, B. Baudry, A. Nicolas, and J.-M. Jézéquel, "Model-driven engineering for software migration in a large industrial context." in *MoDELS*, ser. Lecture Notes in Computer Science, vol. 4735. Springer, 2007, pp. 482–497.

[2] J. Izquierdo and J. Molina, "An architecture-driven modernization tool for calculating metrics," *Software, IEEE*, vol. 27, no. 4, pp. 37–43, 2010.

[3] M. Riley. (2009) Choosing the right tool. [Online]. Available: http://www.oracle.com/partners/campaign/o49field-084396.html. [Accessed: April, 2015]

[4] T. C. Lethbridge and N. Anquetil, "Advances in software engineering," H. Erdogmus and O. Tanir, Eds. New York, NY, USA: Springer-Verlag New York, Inc., 2002, ch. Approaches to Clustering for Program Comprehension and Remodularization, pp. 137–157.

[5] G. Ramalingam and et al., "Semantics-based reverse engineering of object-oriented data models," in *IN PROC. INTL. CONF. ON SOFTWARE ENG.* ACM Press, 2006, pp. 192–201.

[6] R. Bril and et al., "Maintaining a legacy: Towards support at the architectural level," *Journal of Software Maintenance*, vol. 12, no. 3, pp. 143–170, 2000.

[7] O. Sanchez Ramon, J. Sanchez Cuadrado, and J. Garcia Molina, "Model-driven reverse engineering of legacy graphical user interfaces," *Automated Software Engineering*, vol. 21, no. 2, pp. 147–186, 2014.

[8] Composer technologies. Oracle forms to java. [Online]. Available: http://composertechnologies.com/migration-solutions/oracle-forms-to-java/. [Accessed: April, 2015]

[9] VGO Software. Evo. [Online]. Available: http://www.vgosoftware.com/products/evo/walkthrough.php. [Accessed: April, 2015]

[10] Oracle. Jheadstart. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/jheadstart/overview/jhs11-fomrs2adf-overview-130955.pdf. [Accessed: April, 2015]

[11] Pitss. Re-engineering edition-Pitss. [Online]. Available: http://pitss.com/us/products/application-re-engineering-edition/. [Accessed: April, 2015]

[12] Renaps. Ormit. [Online]. Available: http://www.renaps.com/ormit-java-adf.html. [Accessed: April, 2015]

[13] S. Tilley, "Documenting software systems with views vi: Lessons learned from 15 years of research & practice," in *Proceedings of the 27th ACM International Conference on Design of Communication*, ser. SIGDOC '09. New York, NY, USA: ACM, 2009, pp. 239–244.

[14] M. Alalfi, J. Cordy, and T. Dean, "Automated reverse engineering of uml sequence diagrams for dynamic web applications," in *Software Testing, Verification and Validation Workshops, 2009. ICSTW '09. International Conference on*, 2009, pp. 287–294.

[15] G. A. Di Lucca, A. R. Fasolino, and P. Tramontana, "Reverse engineering web applications: the ware approach," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 16, no. 1-2, pp. 71–101, 2004.

[16] T. Richner and S. Ducasse, "Recovering high-level views of object-oriented applications from static and dynamic information," in *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, 1999, pp. 13–22.

[17] E. Duffy and B. Malloy, "A language and platform-independent approach for reverse engineering," in *Third ACIS International Conference on Software Engineering Research, Management and Applications, 2005*, 2005, pp. 415–422.

[18] C. Bennett and et al., "A survey and evaluation of tool features for understanding reverse-engineered sequence diagrams," *J. Softw. Maint. Evol.*, vol. 20, no. 4, pp. 291–315, 2008.

[19] W. Lowe, M. Ericsson, J. Lundberg, T. Panas, and N. Petersson, "Vizzanalyzer - a software comprehension framework," in *Proc. of 3rd Conference on Software Engineering Research and Practise in*, 2003, pp. 127–136.

[20] N. Anquetil and J. Laval, "Legacy software restructuring: Analyzing a concrete case," in *15th European Conference on Software Maintenance and Reengineering*, 2011, pp. 279–286.

[21] S. Mancoridis, B. Mitchell, Y. Chen, and E. Gansner, "Bunch: a clustering tool for the recovery and maintenance of software system structures," in *IEEE International Conference on Software Maintenance*, 1999, pp. 50–59.

[22] M. Girvan and M. E. Newman, "Community structure in social and biological networks." *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[23] Eclipse Community. Eclipse. [Online]. Available: https://eclipse.org/. [Accessed: April, 2015]

[24] Oracle. JDAPI documentation. [Online]. Available: http://www.oracle.com/technetwork/developer-tools/forms/documentation/10g-forms-091309.html. [Accessed: April, 2015]

[25] J. O'Madadhain. JUNG - Java Universal Network/Graph Framework. [Online]. Available: http://jung.sourceforge.net/. [Accessed: April, 2015]

 ISBN: 978-1-61208-438-1

Effects of Recency and Commits Aggregation on Change Guide Method Based on Change History Analysis

Tatsuya Mori†, Anders Mikael Hagward‡,†, Takashi Kobayashi†

† Graduate School of Information Science & Engineering, Tokyo Institute of Technology, 2–12–1 Ookayama, Meguro, Tokyo, Japan

‡ School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden

Email: {tmori, anders, tkobaya}@sa.cs.titech.ac.jp

***Abstract*—To prevent overlooked changes, many studies on change guide, which suggest necessary code changes with using co-change rules extracted from a change history, have been performed. These approaches support developers to find codes that they should change but have not been done yet when they decide to commit their changes. The recommendations by existing approaches are adequately accurate when the tools find candidates. However, these tools often fail to detect candidates of overlooked changes. In this study, we focus on two characteristics to increase the opportunity of recommendation to detect more overlooked changes: one is the consideration of recency, i.e., we use only recent commits for extracting co-change rules, and the other is the aggregation of commits for the same task, i.e., we aggregate consecutive commits fixing the same bug. We investigate the effects of our methods on the quality of co-change rules. Experimental results using typical Open Source Software (OSS) show that the consideration of recency can improve the recommendation performance. Our approach can extract more useful co-change rules and recommend more overlooked changes in a higher rank than without the consideration of recency.**

Keywords–change guide; software repository mining; commit history; software maintenance.

I. INTRODUCTION

As the structure of a software program is scaled up, the effort for the ripple effect analysis [1] significantly increases during maintenance, such as bug correction and implementing new features. For the quality of the product, it is important to complete modifications. To prevent overlooked changes, many change guide methods based on static analysis (SA) have been proposed to analyze the scope of change effects [2]. However, these approaches detect many static dependencies include ones unrelated to the change propagation [3]. Further, SA-based approaches fail to find all of necessary dependencies [4]. It cannot find dependencies between codes and non-code elements, such as configuration files and ones through third party libraries.

To overcome the limitations of SA-based change guide methods, studies focusing on the analysis of a software change history in version control system has been performed. These approaches leverage implicit dependencies (aka. logical coupling [5]) extracted from a change history with data mining techniques.

By using association rule mining, an association between code changes are extracted as rules that indicate "if a file is changed, it is highly possible that another file is changed at the same time". We refer to those rules as *co-change rules*. Zimmermann et al. proposed a co-change recommendation tool, eROSE, that extracts co-change rules from a change history and recommends code elements (e.g., methods or fields) as possible future changes [6]. Their experimental results showed the usefulness of co-change rules for the change guide task. They achieved quite high accuracy for the change recommendation with eROSE. However, the coverage of their recommendation is a few percent; their approach often fails to detect candidates of overlooked changes.

We consider that it is important to expand the coverage for detecting overlooked changes. To increase the opportunity of change guide, we address this low-coverage issue of co-change rules based change guide. In this paper, we propose methods to improve the quality of co-change rules. We focus on two characteristics of change history. One is $rececy$, i.e., how recent the commit was done, and the other is $task$, i.e., which task the commit was related to.

The dependencies that cause change effect become altered along with the project being a long life. That means that the files that had been changed in early term of development might have no dependencies currently. When we use all of the past change histories, we might fail to extract useful dependencies as a consequence of such noise dependencies. We form a hypothesis that we can extract co-change rules strongly related to current changes by considering recency.

When partial changes for a bug fix had been overlooked in past, a developer may have already found these overlooked changes and corrected them. We should treat these change history as a single commit for a bug fix to capture the actual co-change relation. This problem can be generalized as the granularity of commits. Not only for unintended separation, the granularity of commits also depends on the nature of developers and projects. For example, while some developers commit all changes for one task, others may commit changed files in separate revisions for the same task. The difference between developers commits behavior might introduce noise for analysis of change history.

In this study, we investigated effects of the consideration of recency and the aggregation of consecutive commits fixing the same bug on the performance of change guide based on analyzing change history. We formalize our study with the following two research questions:

 ISBN: 978-1-61208-438-1

RQ1 Can we improve the effectiveness of change recommendations with the consideration of recency?

RQ2 Can we improve the effectiveness of change recommendations with the aggregation of consecutive commits fixing the same bug?

The main contributions of this paper are:

- We empirically confirm the usefulness of co-change rules to recommend overlooked changes by using three large OSS projects.
- We indicate that we can recommend more overlooked changes significantly by considering recency as a result of an experiment. Moreover, we can recommend correct overlooked changes in a higher rank than without the consideration of recency.
- We also show that we can improve the performance of recommendation by aggregating consecutive commits fixing the same bug depending on the projects.

Structure of the Paper. Section II discusses the related work. Section III describes the experimental setup. Section IV presents the results of experiments. Section V mentions threats to validity. Section VI closes with conclusion and consequences.

II. Related Work

In this section, we survey the related work in the fields of logical coupling and change guide based on change history analysis. We also survey the related work using some methods that we focus on in our study.

A. Logical Coupling

Gall et al. extracted dependencies between files that have a chance to be changed at the same time by analyzing a change history of a software system stored in version control system, e.g., Concurrent Versions System (CVS) or Git [5]. They called those dependencies "logical coupling." Logical coupling can represent implicit dependency that can not be extracted by static analysis. Lanza et al. proposed Evolution Radar [7]. This tool integrates both file-level and module-level logical coupling information and visualizes those logical couplings. Alali et al. investigated the impact of temporal and spatial locality on the results of computing logical couplings [8]. Wetzlmaier et al. reported insights about logical couplings extracting from the change history of commercial software system [9]. They indicated resulting limitations and recommend further processing and filtering steps to prepare the dependency data for subsequent analysis and measurement activities.

Zimmermann et al. proposed eROSE [6]. This tool extracts method-level logical couplings and recommends code elements as possible future changes. eROSE extracts logical couplings as association rules by association rule mining. Zimmermann et al. performed an experiment to evaluate a performance of recommendations by eROSE in the scenario that "when a developer decides to commit changes to the version control system, can eROSE recommend related changes that have not been done yet?" As a result of the experiment, $Precision$ of recommendations by eROSE was 0.69. That means that 69% of recommendations were correct. The recommendations by eROSE were adequately accurate. However, $Recall$ was 0.023. (Note that they showed $Recall$ was 0.75 in their paper. However, they calculated this value without the ratio of occurrence of their recommendations. We recalculated actual $Recall$ as the product of their $Recall$ and their $Feedback$.) That means that only 2.3% of overlooked changes could be recommended. The performance of recommendations by eROSE is satisfactory, but we are motivated to recommend more overlooked changes.

B. Change guide based on change history analysis

Kagdi et al. proposed sqminer [10]. This tool uncovers the sequences of changed files spuriously and decreases false recommendations. Gerardo et al. showed that Granger causality test can provide logical couplings that are complementary to those extracted by association rules. They built hybrid recommender combining recommendations from association rules and Granger causality. Their experimental results indicated that the recommender can achieve a higher recall than the two single techniques [4].

C. Consideration of Recency

In the field of data mining related to segmentation in direct marketing, Recency, Frequency, and Monetary (RFM) analysis is often performed [11][12]. RFM means how recently a customer has purchased (recency), how often they purchase (frequency), and how much the customer spends (monetary). On the other hand, an association rule mining is often used in the field of change guide for developers, and this method takes only frequency (how often the files are co-changed in the same commit) into account. The dependencies that cause change effect become altered along with the project being a long life, so co-change rules extracted from very old commits might be useless currently. We form a hypothesis that if we also take recency into account for an association rule mining, we can extract co-change rules strongly related to current changes.

D. Aggregation of Commits related to the same task

McIntosh et al. investigated the dependency between source code files and build files. In their research, they aggregated commits related to the same task for an association rule mining to reduce the noise caused by inconsistent developer commit behavior [13]. They aggregated commits based on information extracted from Issue Tracking System and called those aggregated commits "work item". They found that work item is a more suitable level of granularity for identifying co-changing software entities rather than a single commit. An aim of their study is detecting the logical couplings between production code changes and build files. On the other hand, an aim of our study is detecting and recommending overlooked changes using logical couplings between source code files, but we think that we can use the same technique for our study.

III. Experimental Setup

In this section, we describe the tools that we implemented for our experiments, a dataset, experimental settings to address our research questions, and evaluation metrics to evaluate the quality of recommendations.

A. Experimental Environment

We implemented **LCExtractor** for our experiments. LCExtractor extracts co-change rules by using an Apriori algorithm [14]. A co-change rule has a form of "A $\Rightarrow$ B". The notation "A $\Rightarrow$ B" means "if A is changed, it is highly possible that B is changed at the same time." "A" and "B" are called the *left-hand*

 ISBN: 978-1-61208-438-1

TABLE I. HISTORY OF ANALYZED PROJECTS

Project	# Commits	in Git since
Eclipse JDT	21,378	2001–2014
Firefox	395,466	1998–2014
Tomcat	13,824	2006–2014

side and the *right-hand side*, respectively. The left-hand side is a set of files, and the right-hand side is a file. There are some differences between LCExtractor and eROSE. LCExtractor extracts file-level co-change rules, whereas eROSE extracts method-level change rules. eROSE is an Eclipse plugin. On the other hand, LCExtractor is the tool that spuriously makes the situation, where a file that should be changed is overlooked, and evaluate whether LCExtractor can recommend this overlooked file or not. Therefore, LCExtractor can not recommend to developers actually. However, LCExtractor is superior to eROSE in some ways. LCExtractor can track renamed files and deleted files in a change history. Due to this extension, LCExtractor can recommend renamed files using co-change rules extracted from files before renamed, and exclude deleted files from candidate recommendations. LCExtractor extracts co-change rules by analyzing change history in a modern version control system, Git or Subversion, whereas eROSE extracts co-change rules by analyzing change history in CVS.

Let us explain the process of LCExtractor recommendation. We set the range of target commits, e.g., the latest 1,000 commits. LCExtractor extracts co-change rules using older commits before the target commit. LCExtractor spuriously makes the situation, where one file that should be changed is forgotten to commit, by removing a file from the target commit. Finally, LCExtractor recommends files using co-change rules and evaluate those recommendations. LCExtractor performs above processes iteratively for each target commit in order of old to new. Note that LCExtractor uses commits treated as targets in previous iterations for extracting co-change rules.

This tool was executed on an iMac Retina 5k, Late 2014, with a 4GHz Intel Core i7 and 32GB main memory, running Apple OS X Yosemite.

B. Dataset

For our experiments, we analyzed the change history of three large open-source projects (Table I). We cloned all of the commit histories of those projects as of December 2, 2014, from GitHub.

C. Consideration of recency

To investigate how the consideration of recency affects a performance of change recommendations, we compared the case when we used only recent 5,000 commits older than a target commit for extracting co-change rules, to the case when we used all of the commits older than a target commit. We refer to the latter case as a baseline. Concerning Firefox, we used 20,000 commits older than a target commit instead of all of the commits for the baseline. It is because the total number of commits was very large (about 400,000) and it was difficult for LCExtractor to use all of them for calculating co-change rules.

The Apriori algorithm used in LCExtractor required two parameters: minimum support ($minsup$) and minimum confidence ($minconf$). We set $minsup$ to be 0.0025 for Eclipse and Firefox, and 0.001 for Tomcat. We set $minconf$ to be from 0.1 to 0.9 in steps of 0.1 for each project. As described in Section III-A, we need to set the range of target commits. In this experiments, we use 2,000 commits as target commits for Eclipse, 5,000 commits for Firefox, and 3,000 commits for Tomcat.

D. Aggregation of consecutive commits fixing the same bug

To investigate how the aggregation of consecutive commits fixing the same bug affects a performance of change recommendations, we compared the case when we aggregated consecutive commits fixing the same bug, to the case when we did not aggregate. We refer to the latter case as a baseline. In the former case, we referred to a commit message of each commit and checked if the commit message contains a bug id. If the commit message partially matched one of the following regular expressions, we assumed that the commit was fixing a bug.

- bug[# \t]*[0-9]+
- pr[# \t]*[0-9]+
- Show_bug\.cgi\?id=[0-9]+

If the messages of consecutive commits contain the same bug id, we aggregated them, i.e., we treated them as one commit. In our experiment, we did not take other information of commits (e.g., author or an interval between each commit) when we aggregated them.

In this experiment, we used all of the commits older than a target commit for extracting co-change rules, i.e., we did not consider recency. Concerning Firefox, we used 20,000 commits older than a target commit instead of all of the commits as we did in Section III-C. The settings of two parameters ($minsup$ and $minconf$) and the range of target commits were same as Section III-C.

E. Evaluation Metrics

The most important aim of our study is a prevention of overlooked changes. We used an evaluation setting that was similar to the error prevention setting in [6] to evaluate the quality of recommendations. We used $Precision$ and $Recall$ for the metrics of recommendations. $Precision$ represents the accuracy of the recommendations. $Recall$ represents the ratio that the expected files are recommended. Because our aim is a prevention of overlooked changes, $Recall$ is important rather than $Presicion$. In our experiments, the expected recommendation is only one file. As a result of this experimental setting, it is possible that $Precision$ become low unfairly due to many false positives. To evaluate an accuracy of our recommendations, we also use Mean Reciprocal Rank (MRR). This metric is not used in [6]. The high MRR score means that the expected files are recommended in a higher rank. For example, if most of the expected files place top three recommendations, MRR is higher than 0.33. The definition of the metrics for co-change rules is described as follows.

Let the set of co-change rules be $Rule = \{(l_1, r_1), (l_2, r_2), ..., (l_m, r_m)\}$, where l_i is the left-hand set of files and r_i is the right-hand file described in Section III-A. Let commit history be $Commit = \{com_1, com_2, ..., com_n\}$, where com_i is the set of files. For every changed file $c \in com_i$, let $recom_{i,c}$ be the recommended file set from the changed files

without c, as described below. In this experiments, c represents a overlooked change file.

$$recom_{i,c} = \bigcup_{(l_j,r_j)\in Rule} \begin{cases} r_j & (\text{if } l_j \subseteq (com_i - \{c\})) \\ \emptyset & (\text{else}) \end{cases} \quad (1)$$

For every overlooked change file $c \in com_i$, let $rank_{i,c}$ be the rank of {c} in recommendations ranked by $confidence$. The $confidence$ is one of the measure to evaluate the quality of an association rule [15]. If the recommended file set do not contain {c}, $rank_{i,c}$ is 0. Next, we define $feedback_i$, $precision_i$, $recall_i$, and mrr_i for each com_i (note that $|\{c\}|$ is always 1).

$$feedback_i = \frac{1}{|com_i|} \sum_{c\in com_i} \begin{cases} 1 & (\text{if } recom_{i,c} \neq \emptyset) \\ 0 & (\text{else}) \end{cases} \quad (2)$$

$$precision_i = \frac{1}{feedback_i \cdot |com_i|} \sum_{c\in com_i} \frac{|recom_{i,c} \cap \{c\}|}{|recom_{i,c}|} \quad (3)$$

$$recall_i = \frac{1}{|com_i|} \sum_{c\in com_i} \frac{|recom_{i,c} \cap \{c\}|}{|\{c\}|} \quad (4)$$

$$mrr_i = \frac{1}{feedback_i \cdot |com_i|} \sum_{c\in com_i} \frac{1}{rank_{i,c}} \quad (5)$$

Similar to [6], we calculated $precision_i$ with $feedback_i$ as the denominator, in the sense of "the accuracy of when the recommendation was displayed." If $feedback_i$ was 0 (no recommendation is displayed at this commit), we did not calculate $precision_i$ and excluded this commit from calculating $Precision_M$. Unlike [6], we calculated $recall_i$ without $feedback_i$ as the denominator, in the sense of "the rate of detecting overlooked changes for all commits, regardless of whether of not the recommendation is displayed." Similar to $Precision$, if $feedback_i$ was 0, we did not calculate mrr_i and excluded this commit from calculating MRR. Finally, let $Precision_M$, $Recall_M$ and MRR be the average of $precision_i$, $recall_i$ and mrr_i. Additionally, we define the $F\text{-}measure$ by calculating the harmonic mean of $Precision_M$ and $Recall_M$ to evaluate the performance of recommendation comprehensively because there is a trade-off between $Precision$ and $Recall$.

$$Commit^* = \{com_i | com_i \in Commit, feedback_i \neq 0\} \quad (6)$$

$$Precision_M = \frac{1}{|Commit^*|} \sum_{com_i\in Commit^*} precision_i \quad (7)$$

$$Recall_M = \frac{1}{|Commit|} \sum_{com_i\in Commit} recall_i \quad (8)$$

TABLE II. MAXIMUM $F\text{-}MEASURE$

Project	Considering recency	Baseline
Eclipse JDT	0.137 (minconf: 0.5)	0.063 (minconf: 0.5)
Firefox	0.374 (minconf: 0.8)	0.362 (minconf: 0.8)
Tomcat	0.204 (minconf: 0.4)	0.167 (minconf: 0.4)

$$MRR = \frac{1}{|Commit^*|} \sum_{com_i\in Commit^*} mrr_i \quad (9)$$

$$F\text{-}measure = 2 \cdot \frac{Precision_M \cdot Recall_M}{Precision_M + Recall_M} \quad (10)$$

IV. EXPERIMENTAL RESULT

We performed two experiments. In this section, we describe results of each experiment.

A. RQ1: Can we improve the effectiveness of change recommendations with the consideration of recency?

Figure 1 shows the relations between $Precision_M$ and $Recall_M$, and the relations between MRR and $Recall_M$ for each project with varying $minconf$. The red curve represents the case when we consider recency, and the blue one represents a baseline.

Figure 1.(a), Figure 1.(c), and Figure 1.(e) show that $Recall_M$ increased with the consideration of recency although $Precison_M$ slightly decreased in all projects. As we aim to find more overlooked changes rather than to make the recommendations more accurate, that is a good result. Particularly regarding Eclipse, $Recall_M$ significantly increased. In Figure 1.(a), a maximum $Recall_M$ is 0.28 with the consideration of recency whereas a maximum $Recall_M$ of the baseline is 0.11. That means that we could detect 2.5 times more overlooked changes by considering recency than the baseline.

As a result of considering recency, $Precision_M$ is slightly decreased. It is because the number of recommendations increased with consideration of recency, i.e., many false positives decreased $Precision_M$ even if the set of recommendations contained a expected recommendation. However, in the viewpoint of MRR, the recommendations with consideration of recency were more accurate than the baseline. In Figure 1.(b) and Figure 1.(d), MRR clearly increased with consideration of recency. That means that we could recommend overlooked changes in a higher rank with consideration of recency than the baseline. Regarding Tomcat, shown in Figure 1.(f), we could not improve MRR with consideration of recency. We consider that it is because MRR was already high without consideration of recency.

Table II shows a maximum $F\text{-}measure$ of each project in the case of considering recency and a baseline. For all of the projects, a maximum $F\text{-}measure$ achieved by consideration of recency is higher than the baseline. That means that the quality of the recommendations by consideration of recency was comprehensively better than the baseline.

 ISBN: 978-1-61208-438-1

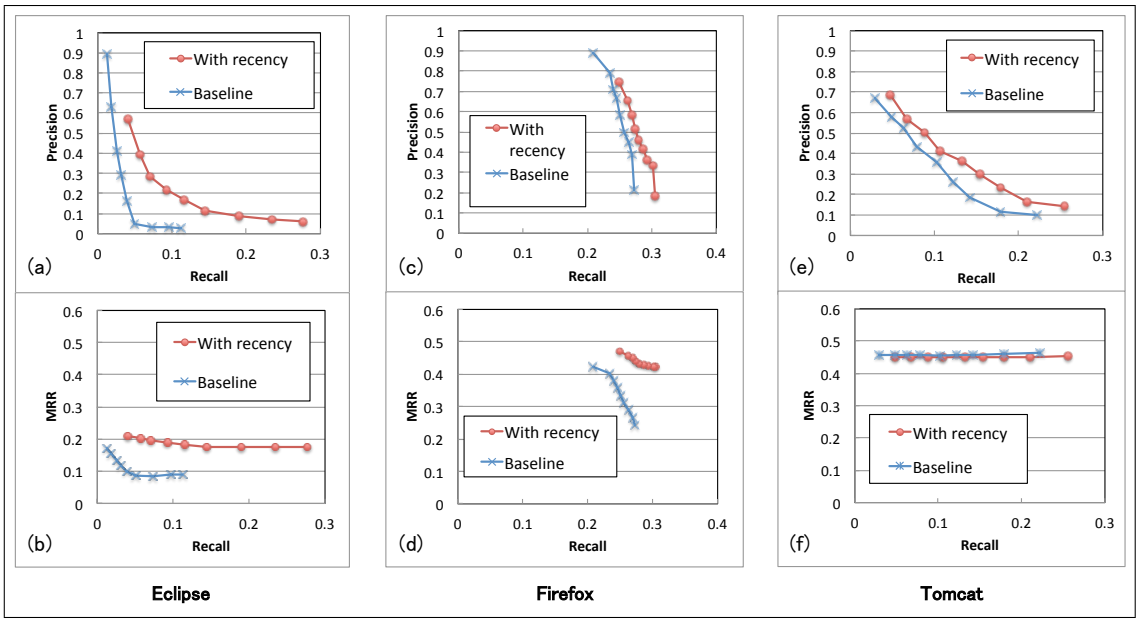


Figure 1. Performance of recommendations by considering of recency and baseline with varying $minconf$.The upper graphs show a relation between $Precision_M$ and $Recall_M$. The lower graphs show a relation between MRR and $Recall_M$.

The answer to RQ1 is **Yes**. If we consider recency, we can extract useful co-change rules that are not able to be extract without consideration of recency. Therefore, $Recall$ of recommendations increase with consideration of recency. Although $Precision$ slightly decrease, we can recommend overlooked changes in a higher rank than without consideration of recency.

B. RQ2: Can we improve the effectiveness of change recommendations by the aggregation of consecutive commits fixing the same bug?

Figure 2 shows the relations between $Precision_M$ and $Recall_M$ for each project with varying $minconf$. The yellow curve represents the case when we aggregate consecutive commits fixing the same bug, and the blue one represents a baseline.

In Figure 2.(a) and Figure 2.(c), we could not improve both $Recall_M$ and $Precision_M$. However, regarding Firefox, shown in Figure 2.(b), $Recall_M$ increased with the aggregation of consecutive commits fixing the same bug. That means that we could extract useful co-change rules that were not able to be extract without aggregation of consecutive commits fixing the same bug depending on projects. As a result of an additional investigation, it is reveal that the number of commits used for extracting co-change rules drastically decreased by aggregating for Firefox (from 20,000 to 15,918), whereas the number of those slightly decreased by aggregating for Eclipse (from 21,378 to 21,098) and Tomcat (from 13,824 to 13,661). We found that the effect that we aggregate consecutive commits fixing the same bug depended on a nature or commit policy of the project.

Table III shows a maximum $F\text{-}measure$ of each project in the case of aggregating consecutive commits fixing the same bug and a baseline. Regarding Firefox, a maximum

TABLE III. MAXIMUM $F\text{-}MEASURE$

Project	Aggregating commits	Baseline
Eclipse JDT	0.063 (minconf: 0.5)	0.063 (minconf: 0.5)
Firefox	0.414 (minconf: 0.8)	0.362 (minconf: 0.8)
Tomcat	0.167 (minconf: 0.4)	0.167 (minconf: 0.4)

$F\text{-}measure$ achieved by aggregation of consecutive commits fixing the same bug is higher than the baseline. Regarding Eclipse and Tomcat, maximum $F\text{-}measure$ of two cases are same because we could not improve both $Recall_M$ and $Precision_M$ as previously described. That means that the quality of co-change rules can be improved by aggregation of consecutive commits fixing the same bug in some cases. Moreover, we also found that aggregation of commits did not affect the performance of recommendation in a negative way.

The answer to RQ2 is, in some cases, **Yes**. If we aggregate consecutive commits fixing the same bug, we can extract useful co-change rules that are not able to be extract without aggregation of commits depending on projects. Even if we can not extract more useful co-change rules by aggregation of commits, there is no harmful effect.

V. THREATS TO VALIDITY

Threats to internal validity relate to errors in LCExtractor and parameter settings. We have carefully checked our code, however still there could be errors that we did not notice. In our experiments, we set $minsup$ to be 0.0025 for Eclipse and Firefox, and 0.001 for Tomcat. It is possible that those values were not appropriate. At the moment, we have no method that decide an appropriate $minsup$ prior to performing experiments.

 ISBN: 978-1-61208-438-1

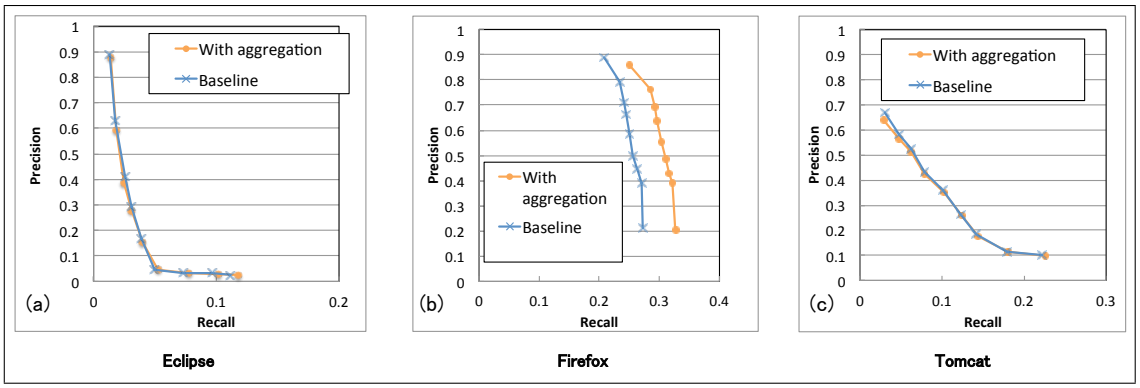


Figure 2. Relation between $Precision_M$ and $Recall_M$ by aggregating of commits and baseline with varying $minconf$.

Threats to external validity relate to the generalizability of our results. We have analyzed 3 different projects. In the future, we plan to reduce this threat further by analyzing more change histories from additional software projects.

Threats to construct validity relate to the experimental settings. We defined using recent 5,000 commits older than a target commit for extracting co-change rules as consideration of recency in the first experiment. However, we did not perform experiments with changing the number of commits for extracting co-change rules. In the future, we plan to reduce this threat further by performing experiments with changing the number of commits used for extracting co-change rules. In the second experiment, we aggregated commits based on only bug id information extracted from commit messages. If we extract more information from Issue Tracking System or Bug Tracking System and use them, we might aggregate commits more appropriately.

VI. Conclusion and Future Work

Numerous studies for supporting developers to find necessary code changes with using co-change rules extracted from the change history have been performed. However, the scope of overlooked changes that existing tools can recommend is limited. In this paper, we focused on the consideration of recency and the aggregation of consecutive commits fixing the same bug. We investigated how they affected the performance of recommendations by using typical OSS (Eclipse, Firefox, and Tomcat). As a result of experiments, we could recommend more overlooked changes by considering recency. We also could recommend correct files in a higher rank than recommendations without consideration of recency. Concerning the case when we aggregated consecutive commits fixing the same bug, we found that the performance of recommendations can be improved depending on projects.

In the future, we plan to perform experiments using more change histories from additional software projects to generalize our theory. In this paper, we indicate that we can improve the performance of recommendations by considering recency. However, we suppose that we can not extract useful co-change rules if we use the small set of commits, e.g., only 10 commits older than a target commit. We plan to investigate how many recent commits are sufficient to extract useful co-change rules.

Acknowledgment

This work is partially supported by the Grant-in-Aid for Scientific Research of MEXT Japan (#24300006, #25730037, #26280021).

References

[1] S. S. Yau, J. S. Collofello, and T. MacGregor, “Ripple effect analysis of software maintenance,” in Proc. COMPSAC’78, pp. 60–65.

[2] L. C. Briand, J. Wust, and H. Lounis, “Using coupling measurement for impact analysis in object-oriented systems,” in Proc. ICSM ’99, pp. 475–482.

[3] M. M. Geipel and F. Schweitzer, “Software change dynamics: evidence from 35 java projects,” in Proc. FSE 2009, pp. 269–272.

[4] G. Canfora, M. Ceccarelli, L. Cerulo, and M. Di Penta, “Using multivariate time series and association rules to detect logical change coupling: an empirical study.” in Proc. ICSM 2010, pp. 1–10.

[5] H. Gall, K. Hajek, and M. Jazayeri, “Detection of logical coupling based on product release history,” in Proc. ICSM ’98, pp. 190–198.

[6] T. Zimmermann, A. Zeller, P. Weissgerber, and S. Diehl, “Mining version histories to guide software changes,” IEEE TSE, vol. 31, no. 6, 2005, pp. 429–445.

[7] M. D’Ambros, M. Lanza, and M. Lungu, “The evolution radar: Visualizing integrated logical coupling information,” in Proc. MSR 2006, pp. 26–32.

[8] A. Alali, B. Bartman, C. D. Newman, and J. I. Maletic, “A preliminary investigation of using age and distance measures in the detection of evolutionary couplings,” in Proc. MSR 2013, pp. 169–172.

[9] T. Wetzlmaier, C. Klammer, and R. Ramler, “Extracting dependencies from software changes: an industry experience report,” in Proc. IWSM-MENSURA 2014, pp. 163–168.

[10] H. Kagdi, S. Yusuf, and J. I. Maletic, “Mining sequences of changed-files from version histories,” in Proc. MSR 2006, pp. 47–53.

[11] P. C. Verhoef, P. N. Spring, J. C. Hoekstra, and P. S. Leeflang, “The commercial use of segmentation and predictive modeling techniques for database marketing in the netherlands,” Decision Support Systems, vol. 34, no. 4, 2003, pp. 471–481.

[12] J. A. McCarty and M. Hastak, “Segmentation approaches in data-mining: A comparison of rfm, chaid, and logistic regression,” Journal of business research, vol. 60, no. 6, 2007, pp. 656–662.

[13] S. McIntosh, B. Adams, T. H. Nguyen, Y. Kamei, and A. E. Hassan, “An empirical study of build maintenance effort,” in Proc. ICSE 2011, pp. 141–150.

[14] P. Bondugula, Implementation and Analysis of Apriori Algorithm for Data Mining. ProQuest, 2006.

[15] R. Agrawal, T. Imieliński, and A. Swami, “Mining association rules between sets of items in large databases,” in ACM SIGMOD Record, vol. 22, no. 2, 1993, pp. 207–216.

 ISBN: 978-1-61208-438-1

Towards Flexible Business Software

Ahmed Elfatatry
Information Technology Department
Alexandria University
Alexandria Egypt
elfatatry@alexu.edu.eg

Abstract—**Software flexibility is a multidimensional problem. Solving one side of the problem might not enhance the situation significantly. This work is motivated by both the problem of software flexibility and the need for a solution for highly volatile business software. The work presented here is based upon ongoing research into software flexibility. The main contribution of this work is the proposal of a new framework to facilitate frequent changes in both the business layer and the presentation layer. Among systems that benefit from such design are workflow systems and document oriented.**

Keywords-Software Flexibility; Document Oriented Systems; presentation layer

I. INTRODUCTION

Software flexibility is the ease with which a software system can be modified in response to changes in system requirements. Software flexibility is a multidimensional problem. Solving one side the problem may not improve the situation significantly. When software is built out of layers, often, applying changes to one layer affects other layers.

Changing one part of a system may require changing a number of related parts; this is known as the "propagation effect" of change. Each of the related parts may need to be dealt with differently. For instance, a change request may affect business rules, user interface, and data. Each of these facets needs to be designed in a way that facilitates change.

The focus of this work is flexibility in business software systems. While all software systems could be subject to change, business software systems are more likely to change as result of their changing environments. Flexibility problems in business systems vary according to the type of the system. Business software systems include business information systems, workflow systems, and document oriented systems [1]. In workflow systems, for instance, modelling techniques produce tightly coupled systems [2]. Minimal change in business requirements may require the change of many parts of a given model. A case in point is the model adopted by the Workflow Management Coalition (WFMC) which embeds transition information within activities [3]. As a result, changing the sequence of activities may require rewriting such activities. Other models integrate business rules within the specification of the activities. This results in activities that are complex and hard to maintain.

A Document-Oriented Application (DOA) is a type of business applications that is built around business documents. User interface in DOAs is both stage-based and role-based where it displays and manipulates business documents in several stages for different roles. Such characteristics bring about a common requirement for applying consistent stage-based and role-based presentation behaviour throughout the entire application.

Adapting DOA after it has been deployed in production usually involves allowing business-experts to change business rules including specifications about stages and/or roles for business documents. Combining this requirement with the stage-based and role-based characteristics brings about a design challenge: the application should be designed to support flexibility both in the business layer and the presentation layer. In other words, the changes made to the business layer should also affect the presentation layer in a consistent manner.

This paper is structured as follows. In Section 2, the problem of building flexible business systems is analysed. Section 3 introduces a framework for dealing with flexibility issues. The evaluation of the proposed work in presented in Section 4. Section 5 discusses the contribution of the work and outlines the future extensions.

II. PROBLEM AND MOTIVATION

Large changes in business requirements naturally lead to large changes in the supporting software systems. When small changes in business requirements lead to large changes in the supporting software system, this indicates the presence of a design problem. In this work, flexibility related problems are classified into two main classes. Each class exposes a different perspective of the system.

A. User Interface problems

An important class of business software is Document Oriented Applications (DOA). A Document Oriented Application is a type of business applications that is built

 ISBN: 978-1-61208-438-1

around business documents. In such systems, work procedures are done by exchanging documents according to some rules related to both the persons using the documents and the state of the given document. A case in point is the exchange of legal documents in a court. Current approaches used in building DOAs fail to solve the issue of reflecting changes in business logic to user interface in a way that retains flexibility [4]. Such approaches have a number of problems discussed below.

• Violating the separation of concerns concept by injecting large crosscutting concerns into user interface [5]. Crosscutting concerns are software features whose implementation is spread across many modules in the form of tangled and scattered code [6]. For example, reflecting presentation behaviour for the active role using current approaches of security architectures results in software that has application code tangled with security code. Such tangling makes it difficult to change security architecture once the software has been deployed [7].

• Concealing the high abstract view of business logic behind presentation changes and blending it within the presentation code. This hardens any attempts to understand or extract business logic that leads to a specific behaviour.

• Producing inflexible solutions that cannot cope with changes in business rules. This leads to DOAs that lose its ability to adapt change once it has been deployed in production. The typical solution to modify or to include new business rules requires a new cycle of development and testing for each modified rule.

• Preventing business-experts who have the required knowledge in a business domain from participating in adapting DOAs. Usually, business experts do not understand programming languages and therefore they cannot directly change the application [8]. Instead, they have to wait for IT-professionals to implement new business rules and to change the behaviour of the user interface.

B. Modelling Problems

Decisions at the conceptual level strongly affect flexibility. The chosen model of decomposition has a direct effect on the cost of change. This sub-section outlines a number of problems that may result from the modelling phase.

• *Inability to respond to frequent changes of business processes*. Most workflow modelling techniques produce tightly coupled systems. A minimal change in a business attribute may require the change of many parts of a given model. For instance, the model adopted by the Workflow Management Coalition [WFMC] embeds transition information within activities [3]. As a result, changing the sequence of activities may lead to rewriting of the activity body itself. Other models integrate business rules within the specification of the activities [9]. Such activities are complex and hard to be maintained**.**

• *Model inconsistency*. The addition or deletion of tasks, relationships, or rules at runtime may cause system inconsistencies especially when changes are done in an ad-hoc manner [10]. Consider a simple order processing where the billing step and the shipping step take place at same time. Assume that a change at run time is made so that the shipping step is performed after the billing step. If at the time of the change, a job had started with shipping, it will never perform the billing step according to the instructions of the new procedure. Thus, a customer will not be billed for the goods that he receives. If there are a large number of jobs being in the same situation at the time of change, then a large number of customers will not be billed. This is a very simple example of a "dynamic bug". Many of these bugs are much more difficult to detect and can have unexpected effects. In the following section, the proposed framework addresses these problems.

C. Research questions

The previous discussion of flexibility problems leads to a number of research questions. First: how can we build user interfaces that can accommodate changes in other layers of the software system? Second: how can workflow systems be more adaptive to change?

III. THE PROPOSED FRAMEWORK

To address the issues described above, we propose a framework for flexibility. The following sub sections describe the proposed framework.

D. Conceptual view

The proposed framework defines a workflow as a set of activities as shown in Figure 1. The upper part of the figure shows a design time view of a workflow. The lower part of the figure shows the runtime view of the figure. A workflow consists of one or more activities ordered according to some transition flow rules. Transition flows are not embedded within activities. They are modelled as first class entities. Each activity is assigned to a specific role according to binding conditions. Role binding rules postpone the assignment of an activity to an available user until runtime [11].

At runtime, activities are bounded to the appropriate services through service requests. Business rules can be bound to workflow at any time during its life cycle, providing the ability to customize the workflow while it is executed.

 ISBN: 978-1-61208-438-1

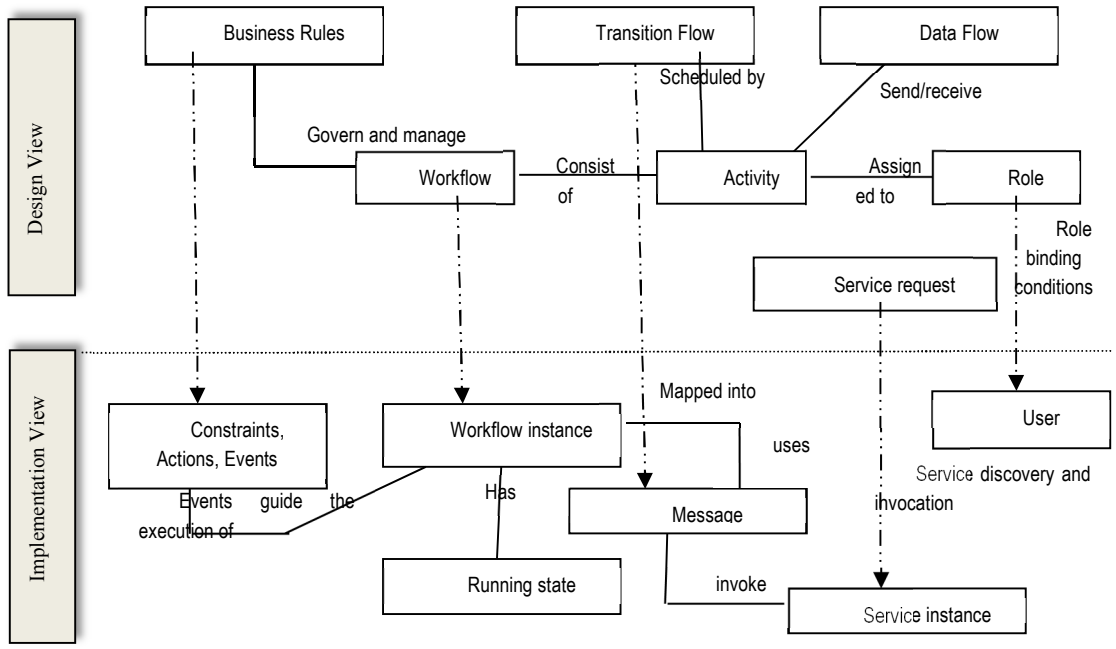


Figure 1. Design View & Implementation View

E. 3.2 Presentation Behaviour Layer (PBL)

In typical DOAs, a system is divided into three layers: Data-Access layer, Business layer and Presentation layer. In the proposed approach, we introduce a fourth layer: Presentation-Behaviour Layer (PBL) as shown in Figure 2. The main goal of this layer is to provide a mechanism for applying presentation changes in a consistent manner.

The PBL externalizes the logic of applying presentation-behaviours instead of hard coding it within the presentation layer. This externalization provides support for building flexible DOAs. The PBL consists of (PBM) and Presentation-Behaviours Run-time (PBR). The PBM is responsible for defining and storing presentation behaviours, while the PBR is the responsible for applying such behaviours during the runtime. The arrows show that PL uses services from BL and BL uses services from DAL. Arrows on the left, show the interaction between PBL and PL in response to a given change.

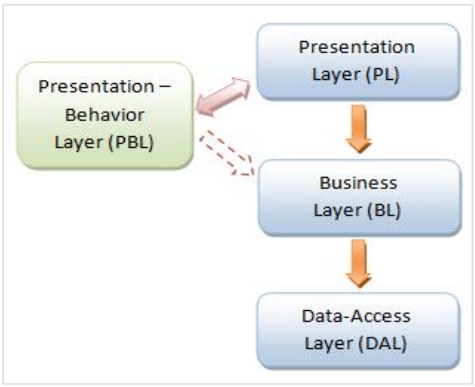


Figure 2. Presentation Behavior Layer

Presentation-Behaviour Model (PBM). The PBM consists of state machines and sequence flows. Each state diagram describes the behaviours that the system should apply at each stage of the process. One of the main objectives of PBM is to externalize and store full specifications about presentation changes outside the presentation code. The specifications are stored in XML documents which contain all the information required to describe how and when to apply presentation behaviours. When a change happens, it is analysed to its atomic element and then reflected to the presentation behaviour layer.

State machines. State machines are the ideal placeholders to store specifications about presentation behaviour for each process stage. They are suitable for representing the stages of business documents. In contrast to other approaches that blend presentation behaviour within the source code, the state diagrams keep the original definition of these behaviours inside the BPM model. Obviously, this simplifies the understanding of business rules that lead to a specific presentation behaviour. In addition, storing presentation behaviours in state diagrams representations rather than source code allows business-experts to participate in the development process by defining presentation behaviours for each business requirement.

In the proposed approach, we employ state diagrams to store specifications about business processes and their related presentation behaviours. Therefore, we need to store extra specifications about presentation behaviours for each combination of a stage and a role.

State: a state corresponds to a document stage in a business process. Usually the state identifies a significant point in the lifecycle of a business process.

Actions: an action represents a business logic that should run to perform a business task. In our approach, actions are modelled as sequence diagrams which

 ISBN: 978-1-61208-438-1

provide simplicity and flexibility. Operations and Transitions are concrete forms of actions. From the user interface perspective, actions (Operations and Transitions) are reflected to user interfaces as tasks that can be triggered by end-users.

Transitions: a transition represents a change in the document stage. The transition connects a source to target state. At any given time only one transition can be executed for each document.

Guard conditions: a guard condition is an optional specification that describes business rules. It has to be evaluated before a transition can be executed.

Operations: an operation represents a business logic that should run to perform a business task. Operations can range from simple and common actions such as CRUD (Create, Retrieve, Update, and Delete) operations, to complex and custom tasks such as "Calculating Taxes".

Attributes: an attribute represents a document element that can be entered, modified and displayed. The concept of attributes is introduced to the proposed state diagrams to allow presentation behaviours to be defined at the granularity of attributes.

Roles: the role-based nature of business documents requires proper communication with access control model. In the proposed approach, we enriched state diagrams to define access controls for each element in each stage.

Specifying Presentation Behaviour. The proposed state machines have additional attributes that describe presentation behaviour. The objective of these attributes is to provide specifications that allow PBR to apply presentation changes automatically to user interfaces. The additional attributes deal with the following issues.

- **Controlling tasks.** User interfaces in document oriented applications provide end-users with a set of tasks that are appropriate for both active stage and role. Storing specifications about such tasks allows PBR to display proper tasks upon each stage change. Definitions of tasks include both visual and functional aspects. These specifications transform the tasks from being code-oriented to a higher and more abstract form. Such form is more business-expert oriented. It treats tasks as standalone elements that can be granted to or denied to certain roles.

- **Controlling default presentation modes and exceptions.** A document stage usually defines whether the user interfaces allow end-users to modify document information or not. The default mode allows readers to easily figure-out the expected behaviour especially in user interfaces that represent documents with large set of attributes.

- **Controlling common handlers.** The architecture of business documents results in common and redundant operations that could be applied to any document instance. For instance, all business documents provide common business operations such as CRUD operations, validation handlers, state transitions and etc. Although these operations are usually written centrally in the data access layer (DAL) and the business layer (BL) respectively, however, the code that calls them and displays their results to end-users is usually written in each user interface. Externalizing the decisions to activate or deactivate such common operations into the definitions of state machines provides more flexibility to adapt user interfaces according to the characteristics of each document stage.

- **Controlling default authorization mode and its exceptions.** Similar to the presentation mode, the default authorization mode simplifies defining authorizations to document information.

- **Controlling role access**. Although the default authorization mode discussed above facilitates the definitions of implicit authorizations, however, there is a need in some situations to define access roles in the granularity of attributes, transitions, and operations. We believe that this part is the most complex and is responsible for most of the crosscutting code.

IV. EVALUATION

At the architectural level, software quality attributes such as flexibility are hard to measure using direct quantitative measures. Other indirect methods are more suitable for the nature of this work. Two methods have been adopted to evaluate this work. The first method examines the effects of different types of changes on the proposed system and compares the results to those of traditional workflow systems. The second method evaluates this work by cross-referencing the features of this solution and a number of flexibility requirements.

A. Comparing the proposed framework with related work

One way to measure the success of the proposed solution to achieve flexibility is to test it on different scenarios of change and compare the ease of change with the results of traditional workflow management systems.

A common area of change in businesses is policy change. Policy changes usually have a substantial effect on workflows. Existing workflow models deal with business policies and rules in different ways. Usually, workflow systems introduce only a limited type of constraint that could be defined within an activity as a transition condition. Modeling business policies with such a model will be very hard. It may only be modeled as a new activity with different behavior, and different pre and post conditions which leads to a complex design.

Another way to model policies is to use a rule based workflow model. The entire workflow composition logic is specified in the form of if/then rules. Such a model determines the boundaries of a workflow, and leaves the freedom to the designer to specify the transitions between the activities. The workflow components such as activities, flows, roles, business policies are expressed in terms of activities built in process specification. This results in processes that are not modular, complex, and hard to maintain. In such a case, business rules are hard to change without affecting the core composition of the model. This way of modeling decreases the flexibility of the workflow.

 ISBN: 978-1-61208-438-1

The Proposed model introduces rules as a first class abstraction that governs and guides workflow execution. Each rule has enforcement conditions which state when and how such a rule is enforced inside the flow. Rules are not embedded within processes. Change in policies is enforced by changing related rules. This principle makes the workflow more simple and easy to maintain. Workflow enactment engine enforces policies by checking rules related to each step before performing it. Rules do not only govern activities but also govern role binding, services specifications, and exception handling.

The Model-View-Controller (MVC) is a software pattern for implementing the separation of concerns concept in the implementation of software systems. The work presented here focuses on providing a mechanism for reflecting changes on the presentation layer specifically.

SNATA defines service oriented architecture for N-tier application [11], however, it does not provide a mechanism for change propagation between layers.

B. *Matching the features of the solution to the specified flexibility requirements*

The proposed solution has been evaluated against a set of flexibility requirements. This set of requirements is derived from a number of well-established software engineering principles such as abstraction, separation of concerns, and loose coupling. The requirements are discussed below.

R1: Support model evolution. Evolution of workflows occurs over time as a result of changing tasks, priorities, responsibilities, and people. Modifications should be allowed at design time as well as at runtime. The proposed solution allows structural changes as well as behavioural changes. Structural changes allow model evolution. The Rule manager provides an interface to accomplish this requirement.

R2: Allow function/provider decoupling. The provider of a specific functionality may not be specified until runtime. Hard coding such information at design time leads to systems that are not flexible. In the proposed solution, activities are implemented as services. Services are selected according to some criteria that may not be known until runtime. Service selection constraints are sent through service requests to each running instance to select a suitable service and source of provision. A new activity or behaviour could be added at runtime to allow composition of a complex task.

R3: A workflow has to provide an integrated multiple view of a business system.
A workflow model has to provide high level of abstraction, and support visualization of its parts. The Proposed framework combines an activity based model, role model and a rule based model. A business system may be viewed from one or more perspectives: roles, processes, or rules. The proposed framework provides a multi-view modeling of a business system.

R4: Support the management of evolving workflow schema. Changes in business environment have to propagate to running workflow instances. A robust management system has to support propagation of change to running instances in a consistent way. The presented work didn't address this requirement.

V. CONCLUSION

The main contribution of this work is the introduction of a framework for dealing with change in business software. The focus is on workflow systems and user interface in document oriented systems.

A major drawback of current approaches for building document oriented applications is neglecting the impact of change in business rules on user interfaces. The result is having systems that are hard to change when business requirements change. While it may be easy to change the code related to business rules, the impact of such changes on the user interface may cause undesirable knock-on effect. For instance, many researches focus on how to provide flexibility in the business layer by providing workflow based solutions. However, the impact of such changes on user interface is usually ignored.

It is necessary that flexibility should be addressed in each logical layer and also between different communicating layers. That is why it is common that many business applications that provide flexibility in the business layer and also provide flexibility in presentation layer fail to sustain flexibility across the boundary between the two layers.

To address such problems, we introduced the Presentation Behaviour Layer (PBL) as a solution of providing flexibility between business layer and presentation layer. We believe that, the PBL can eliminate most of the crosscutting concerns usually found in document oriented applications to apply presentation changes while keeping flexibility. In addition, the visual representation of PBMs allows business-experts to modify their applications based on business rules without the need to touch the source code.

Building flexible workflow systems comes at a cost. The main cost is the implementation efficiency. While separating roles, business rules, and invocation conditions, leads to a flexible design, it certainly adds processing overhead.

Although a complete analysis of flexibility problems and limitations has been discussed, the proposed solution has mainly focused on modelling problems. Runtime limitations still need more research. Currently, we are working on enhancing the performance of workflow engines. The ongoing work focuses on the development of more propagation strategies and building workflow engines able to efficiently weave rules with activities.

Three medium sized companies with average of seven developers each have been chosen to implement the proposed framework. The framework will be applied to existing systems that are subject to frequent change requests. A comparison between the performance before and after using the framework will be published later.

REFERENCES

[1] C. Wiehr, N. Aquino, K. Breiner, M. Seissler and G. Meixner, "Improving the flexibility of model transformations in the model-based development of interactive systems," in *Proceedings of the 13th IFIP TC 13 international conference on Human-computer*

interaction - Volume Part IV, Lisbon, Portugal, 2011,pp. 540-543.

[2] S. Bhiri, G. Khaled , O. Perrin and C. Godart, Overview of Transactional Patterns: Combining Workflow Flexibility and Transactional Reliability for Composite Web Services, Springer Berlin / Heidelberg, 2005, pp. 440-445.

[3] WfMC, "Interface 1: Process Definition Interchange," [Online]. [Accessed May 2015].

[4] O. Chapuis, D. Phillips and N. Roussel, "User interface façades: towards fully adaptable user interfaces," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*, Montreux, Switzerland, 2006, pp 309-318.

[5] A. Marot, "reserving the separation of concerns while composing aspects on shared joinpoints," in *4th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2009*, Orlando, Florida, USA., 2009, pp. 837-839.

[6] A. Sabas, S. Shankar, V. Wiels and M. Boyer, "Undesirable Aspect Interactions: A Prevention Policy," in *Theoretical Aspects of Software Engineering, Joint IEEE/IFIP Symposium*, Montreal, Montreal, QC, Canada, 2011, , pp. 225-228.

[7] G. Chao, "Human-Machine Interface: Design Principles of Visual Information in Human-Machine Interface Design," in *IHMSC '09 Proceedings of the 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*, IEEE Computer Society Washington, 2009, pp. 262-265.

[8] M. Mike and D. Dwight , "End user developer: friend or foe?," *J. Comput. Small Coll.,* vol. 24, no. 4, pp. 40-45, April 2009, pp. 42-49, 2009.

[9] T. Sterling and D. Stark, "A High-Performance Computing Forecast: Partly Cloudy," *Computing in Science and Eng.,* vol. 11, no. 4, pp. 42-49, 2009.

[10] M. Blake, A. Bansal and S. Kona, "Workflow composition of service level agreements for web services," *Decision Support Systems,* vol. 53, no. 1, April 2012, pp. p. 234–244,.

[11] A. Elfatatry, Z. Mohamed and M. Eleskandarany, "Enhancing Flexibility of Workflow Systems," *80 Int.J. of Software Engineering, IJSE,* vol. 3, no. 1, pp. 79-92, 2010.

[12] T. . C. Shan and W. H. Winnie , "Solution Architecture for N-Tier Applications," in *Proceedings of the IEEE International Conference on Services Computing*, September 2006, pp. 234-244.

[13] C. Ackermann, M. Lindvall and G. Dennis, "Redesign for Flexibility and Maintainability: A Case Study," in *Software Maintenance and Reengineering*, Kaiserslautern, Germany, 2009, 2009, pp. 259-262.

[14] A. Bruno, F. Patern and C. Santoro, "Supporting interactive workflow systems through graphical web interfaces and interactive simulators," in *TAMODIA '05 Proceedings of the 4th international workshop on Task models and diagrams*, Gdansk, Poland, 2005, pp 63-70.

[15] D. Gaurav, "A survey on guiding logic for automatic user interface generation," in *Proceedings of the 6th international conference on Universal access in human-computer interaction: design for all and eInclusion - Volume Part I*, Orlando, FL, 2011, pp. 365-372.

[16] IEEE, "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 610.12-1990,* pp. 1-84, Dec 1990.

 ISBN: 978-1-61208-438-1

EBGSD: Emergence-Based Generative Software Development

Mahdi Mostafazadeh, Mohammad Reza Besharati, Raman Ramsin
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
e-mail: {mmostafazadeh, besharati}@ce.sharif.edu, ramsin@sharif.edu

***Abstract*—Generative Software Development (GSD) is an area of research that aims at increasing the level of productivity of software development processes. Despite widespread research on GSD approaches, deficiencies such as impracticability/impracticality, limited generation power, and inadequate support for complexity management have prevented them from achieving an ideal level of generativity. We propose a GSD approach based on a novel modeling paradigm called 'Ivy'. Ivy models the context domain as a set of conceptual phenomena, and depicts how these phenomena emerge from one another. Our proposed approach, Emergence-Based Generative Software Development (EBGSD), uses Ivy models for modeling how a software system (as a phenomenon) can emerge from its underlying phenomena, and can provide an effective means for managing software complexity. Developers can also elicit generative patterns from Ivy models and utilize them to increase the level of reuse and generativity, and thus improve their productivity.**

Keywords-generative software development; phenomenon; emergence; conceptual model

I. INTRODUCTION

As Mens points out, "Software systems are among the most intellectually complex artifacts ever created by humans" [1]. Managing software complexity is indeed the main impetus behind many research areas in software engineering. Generative Software Development (GSD) aims to address this issue through increasing the level of automation in software development, which also enhances productivity. Despite widespread research on GSD approaches such as Model-Driven Development (MDD), Software Product Lines (SPL), Program Development from Formal Specifications, Generative Patterns, and High-Level Programming Languages, there are certain disadvantages in each of them that have prevented researchers from achieving an ideal level of generativity in software development. For instance, in Czarnecki's GSD approach [2], two methods (Configuration and Transformation [3]) have been suggested for transition from the problem domain to the solution domain; although this approach is well-established, it has not achieved an ideal level of generativity, mainly due to deficiency in generation power, inflexibility of configuration, over-abstractness, inattention to seamlessness, and ambiguities in transformation. Furthermore, some of the approaches, such as MDD and High-Level Programming Languages, are deficient as to their support for complexity management. These shortcomings (further explained in Section II) are the main motivations behind this research.

We propose a GSD approach based on a novel modeling paradigm called *Ivy*, originally proposed by Besharati in a seminar report in 2013 [9]. *Phenomenon* and *Emergence* [13] are the two basic concepts of the Ivy paradigm. The Ivy paradigm prescribes a way for modeling the emergence of a conceptual phenomenon from its underlying phenomena. Emergence is recursive: an Ivy model takes the form of a digraph that shows how a phenomenon emerges from its underlying phenomena, which in turn emerge from other phenomena, and so on.

In the Ivy-based software development approach that we propose herein (which we have chosen to call Emergence-Based Generative Software Development, or EBGSD for short), the target software system is considered as a phenomenon that emerges from its underlying phenomena, and is therefore represented as an Ivy model. The Ivy model helps manage the inherent complexity of software systems. Furthermore, it is possible to extract generative patterns from Ivy models and utilize them to increase the level of reuse in software development processes, and thereby promote generativity. The evolutionary nature of the modeling approach makes it highly practical, and can lead to a high level of flexibility in software development. We have also proposed a methodology for applying EBGSD to real-world projects. EBGSD promotes seamlessness, and can improve software processes as to smoothness of transition among development activities.

The rest of the paper is structured as follows: Section II provides an overview of the research background through focusing on a number of prominent GSD approaches; in Section III, we introduce the Ivy modeling paradigm as the basis for our proposed approach; our EBGSD approach and its corresponding methodology are proposed in Sections IV and V, respectively; an illustrative example of the application of EBGSD is given in Section VI; finally, Section VII presents the conclusions and suggests ways for furthering this research.

II. RESEARCH BACKGROUND

Software generation is an old ideal that has been pursued and evolved over decades. The advent of programming languages and compilers can be considered as the first step towards enhanced productivity in software development. The field has evolved over decades: for instance, in the context of MDD, programming languages and compilers have been replaced by Domain-Specific Languages (DSLs) and model/code generators. Due to the vastness of the research conducted on software generativity, it is not possible to discuss all of them here; hence, we will focus on the four most prominent approaches, as listed below. Our main purpose in this section is to demonstrate the motivations for this research, and to outline the research objective.

 ISBN: 978-1-61208-438-1

Genetic and evolutionary approaches: these approaches aim at generating complex systems through creation of a simple generative system to generate new constructions that ultimately lead to the desired complex system [4]. The main problem with these approaches is that due to their high level of inherent randomness, they are not applicable to systems with specific requirements.

MDD: MDD considers models as first-degree entities that drive the software development process and serve as the basis for generating the target software [5]. In this approach, software is developed through creation of models at a high-level of abstraction, and then transformation of these models into their lower-level counterparts (and ultimately software) based on certain mappings. Although this approach has become popular in recent years, there are major problems that prevent it from achieving an ideal level of automation. For instance, although this approach intends to reduce software complexity, it in fact just shifts the complexity [6]: development is easy and straightforward when the modeling levels and their corresponding mappings have been specified, but defining the levels and the mappings themselves is by no means straightforward.

SPL: in the software product line approach, instead of developing a single software system from scratch, the focus is on a family of systems that are developed from a set of common reusable components by applying a defined process [7]. To be more precise, a software product line is a set of software-intensive systems that share a common set of features, and that are developed from a common set of core assets [8]. As implied by this definition, SPL aims to improve the productivity of software development processes through providing a higher level of reuse; but the definition makes no hint of any automation involved in the process. Hence, SPL has not been able to achieve an ideal level of generativity. Moreover, creating reusable assets is a costly process, which might even adversely affect the productivity of software development processes.

Czarnecki's GSD approach: similar to SPL, Czarnecki's approach aims at increasing the productivity of software development processes through focusing on families of systems [2]. The main difference between this approach and the SPL approach is that it emphasizes automated composition of components, whereas manual composition is acceptable in SPL. However, just like SPL, GSD too can have an adverse affect on productivity.

As observed in the above approaches, although they have strived to increase the level of software generativity, certain deficiencies prevent them from achieving the ideal level of generativity in software development, and overcoming these deficiencies is the objective of this research. Specifically, genetic approaches enjoy a high level of automation, but are not practicable. On the other hand, MDD, SPL, and GSD are practicable, but are deficient as to complexity management, automation, and productivity; to be precise, these approaches just replace development complexity with mapping complexity.

III. IVY PARADIGM

Ivy [9] is a modeling paradigm for representing conceptual phenomena and their emergence. Conceptual phenomena are typically regarded as abstractions of real-world phenomena. Ivy is based on the notion that conceptual phenomena can be combined, and a new conceptual phenomenon thus emerges. We model this fact in the *Ivy Model*; as seen in Figure 1, an Ivy model is a directed graph in which nodes represent phenomena, and arcs represent emergences. As an example, consider the following three conceptual phenomena: *car*, *red*, and *wheel*, which are the results of abstraction from their real-world counterparts. As shown in Figure 1, from a certain point of view, the phenomena *car* and *red* can be combined, and the phenomenon *red car* thus emerges. From another point of view, the phenomena *wheel* and *red* can be combined, and the phenomenon *red wheel* emerges. The phenomena *car* and *red wheel* can be combined, and from two different points of view, two phenomena emerge: *red-wheeled car*, and *red car wheel*.

The world of software development is full of representation, combination and emergence of conceptual phenomena. Requirements engineering is concerned with conceptual phenomena directly abstracted from real-world phenomena. Some of these phenomena are combined, and other conceptual phenomena emerge as a result. For instance, the conceptual phenomena *actors*, *use cases* and their *relationships* are combined and the phenomenon *use case diagram* emerges; or in goal-oriented requirements engineering, certain phenomena (i.e., *goals)* could be combined, and a higher-level goal would emerge. Software *platforms* are themselves conceptual phenomena that emerge from other phenomena (e.g., *requirements*). Design and implementation phases are concerned with combination of *requirement* and *platform* phenomena and the emergence of *software-solution* phenomena.

Since the dependencies in an Ivy model are unidirectional, it can enhance understandability and modifiability, leading to better complexity management. The Ivy model may look very similar to other models such as goal models [10] and feature models [11], but there are fundamental differences. In those models, relationships have very specific semantics: in goal models, relationships mean *Why* and *How* [10], and in feature models, relationships show the semantics of *Has* [12]. Whereas in Ivy, the emergence relationship has a general meaning, and its concrete semantics depends on the perspective upon which it is based. The semantic generality of emergence is an important feature of Ivy, which makes it capable of tying all conceptual phenomena together. Thus, the relationships in feature models and goal models can be considered as special kinds of emergence.

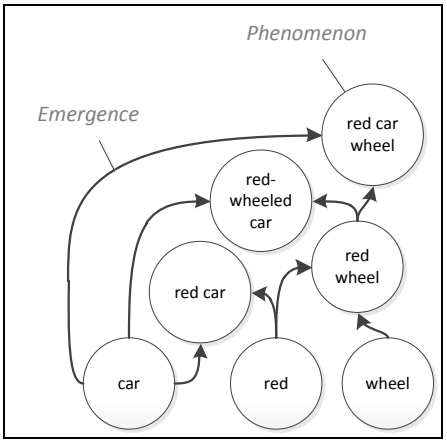


Figure 1. Example of an Ivy model.

In the next section, we will explain our proposed GSD approach based on the Ivy model.

IV. EBGSD APPROACH

In our proposed Ivy-based software development approach, EBGSD, the system for which a software solution is required is considered as a problem-domain phenomenon, which itself emerges from lower-level phenomena. We represent this fact in an Ivy model. This process is applied recursively: for each phenomenon at every level, we can represent the emergence of that phenomenon from lower-level phenomena. Although this process is theoretically endless, we continue it until the phenomena at the lowest level can be considered as basic phenomena in the problem domain; these 'leaf' phenomena are the finest-grained phenomena required for fulfilling the purposes of the developers. Similarly, we consider the software platform (solution domain) as a phenomenon, and draw an Ivy model to represent its emergence. In the next step, we combine the two Ivy models (problem-domain and solution-domain), and software-solution phenomena emerge. The combination process is initiated in a manual fashion, but it can then proceed with a certain degree of automation through producing and applying *Ivy generators*. This is done by identifying recurring and reusable patterns of combination, and capturing them as Ivy generation patterns. An Ivy generator can then be developed to automatically apply these patterns, and thereby combine the two source models (problem-domain and solution-domain) into the destination model (software-solution).

Ultimately, the code corresponding to each software-solution phenomenon is produced. To this aim, it is first determined which underlying software solution phenomena affect the generation or modification of the code corresponding to the target phenomenon; a set of *code generators* are then developed for these underlying phenomena, the outputs of which should be conveyed to the codes of higher-level phenomena. Many of these code generators are typically reusable, and therefore act as patterns. The generation logic embodied in the code generator of each phenomenon utilizes the outputs of lower-level generators (i.e., the codes of the corresponding lower-level phenomena) to generate the code of the target phenomenon. Examples of the abovementioned models and patterns are provided in Section VI.

In the next section, we propose an iterative-incremental methodology for applying EBGSD in software development projects.

V. A METHODOLOGY FOR APPLYING EBGSD

The iterative-incremental software development methodology hereby proposed for applying EBGSD consists of five iterative workflows (as shown in Figure 2): 1) Production of Problem-Domain Ivy Model, 2) Production of Software-Platform Ivy Model, 5) Emergence of Software-Solution Ivy Model, 4) Production of Ivy Generators, and 5) Production and Application of Code Generators. These workflows are iterated in order to gradually produce the models and the target system. This methodology is not a full-lifecycle process; it should be augmented with complementary activities (including umbrella activities and post-implementation activities) in order to become practicable. The workflows will be explained throughout the rest of this section. Section VI provides examples of the products of these workflows.

A. Production of Problem-Domain Ivy Model

We consider the system (for which we intend to develop software) as a phenomenon, and draw an Ivy model depicting the phenomena from which it emerges. Requirements and structural constituents of the problem domain are considered as important phenomena in this model. As previously mentioned, based on different points of view, different Ivy models can be produced for the same purpose. Drawing the Ivy model requires no special skills on the part of the modeler; developers can draw their own based on their particular perspectives of the system. For example, an analyst who knows how to model the requirements as use cases can regard each use case as a phenomenon emerging from its steps, and each step as a phenomenon emerging from the phenomena in the structural view of the system. It should be noted that since everything is represented as phenomena, it is necessary to add certain semantic phenomena in order to provide the readers and the generators with adequate semantics. For example, if *Add Student* is a use case (represented as a phenomenon of the same name), it is necessary to represent the emergence of this phenomenon from a phenomenon named *Use Case*.

B. Production of Software-Platform Ivy Model

We consider the software platform (solution domain) as a phenomenon, and draw an Ivy model depicting the phenomena from which it emerges. For example, in the object-oriented platform, the phenomenon *Class* emerges from the phenomena *Attribute* and *Method*, the phenomenon *Attribute* itself emerges from its *Type*, and so on. In some cases, it is enough to just model the solution-domain phenomena; emergences are left out in such cases.

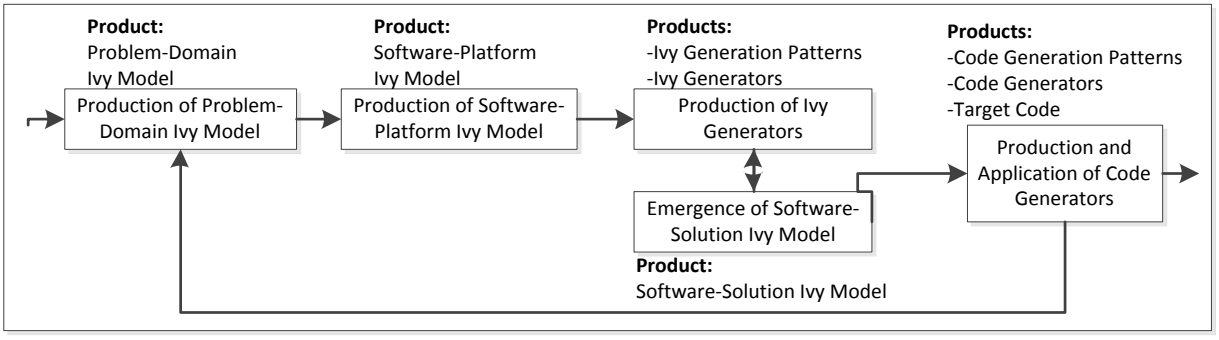


Figure 2. A methodology for applying EBGSD (workflows and products).

C. Emergence of Software-Solution Ivy Model

The problem-domain and software-platform Ivy models are combined (partly manually, and mostly by applying the Ivy generators produced in the next workflow), and the software-solution Ivy model emerges; as this workflow is dependent on Ivy generators, it generally overlaps with the next workflow. It should be noted that three types of phenomena may affect the emergence of each software-solution phenomenon: problem-domain phenomena, solution-domain phenomena, and other software-solution phenomena. For instance, the software-solution phenomenon *Student* emerges from other software-solution phenomena (*Name* and *Age*), as well as a solution-domain phenomenon (*Class*) and its problem-domain counterpart (*Student*).

D. Production of Ivy Generators

Based on the problem-domain and software-platform Ivy models, a set of Ivy generation patterns are elicited and their corresponding Ivy generators are developed (to be updated iteratively). This workflow starts after combination patterns have been identified through manual combination of the source Ivy models, and its results are in turn used for producing the software-solution Ivy model; it therefore overlaps with the previous workflow.

E. Production and Application of Code Generators

Based on the software-solution Ivy model, a set of code generators are developed (as explained in Section IV). These generators are in fact responsible for realizing what the literature on emergence calls *Radical Novelty* [13]: unpredicted and rich features that cannot be anticipated until they actually surface.

VI. EXAMPLE

A partial problem-domain Ivy model for an education system is illustrated in Figure 3. The Ivy model has been drawn based on two different points of view. The left part of the model is drawn based on a structural view of the system. *School* and *Student* are two pivotal entities in the system, so we have represented them as two phenomena that have both emerged from the *Entity* phenomenon. The **Bold Tags** shown on some of the phenomena indicate the emergence of those phenomena from a phenomenon with the same name as the tag; hence, there is no need to explicitly show the corresponding emergence arcs, and excessive complexity is thereby avoided. For instance, the **Entity** tag on the *Student* phenomenon is equivalent to an emergence arc from the *Entity* phenomenon to the *Student* phenomenon. The PD prefix means that the phenomenon belongs to the problem domain. One important relation in the system is the relation between a school and its students; hence, we have represented it as a phenomenon that has emerged from three phenomena: *School*, *Students*, and *Relation*.

The right half of the Ivy model is drawn based on a functional view of the system. One of the system's use cases (*Add Student*) has been represented as a phenomenon, emerging from its *Steps* and the *Use Case* phenomenon. The use case steps themselves have emerged from structural-view phenomena and certain semantic phenomena such as *Add* and *Command;* these semantic phenomena are essential for developing code generators. It should be noted that these points of view are chosen from among many possible alternatives; developers draw the Ivy model based on their own perspectives (e.g., a feature-driven point of view). Each phenomenon and emergence itself may possess implicit semantics, which can be represented separately as an Ivy model; however, due to practicality considerations, the process of Ivy modeling should be brought to an end before the complexity becomes pointlessly overwhelming.

A partial solution-domain Ivy model for the object-oriented platform is shown in Figure 4. This Ivy model has been produced based on well-established object-oriented notions, e.g., a class is an encapsulation of certain attributes and methods. It should be noted that solution domains are typically application-independent.

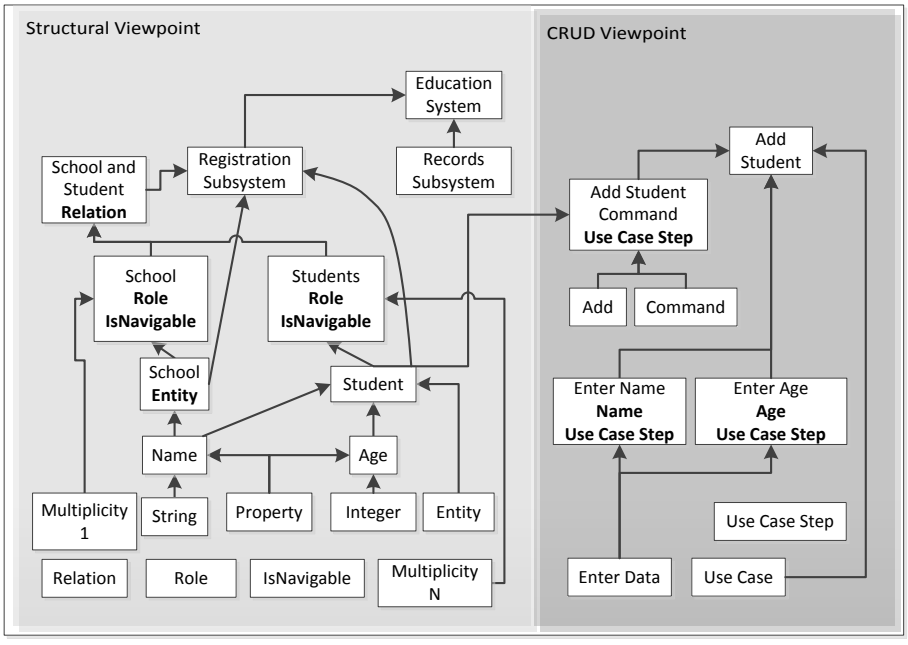


Figure 3. Partial problem-domain Ivy model for an education system.

 ISBN: 978-1-61208-438-1

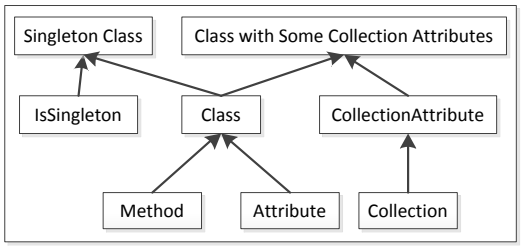


Figure 4. Partial solution-domain Ivy model for the object-oriented platform.

A partial software-solution Ivy model for the education system is illustrated in Figure 5. This model is obtained through extracting and applying a set of Ivy generation patterns. These patterns are illustrated in Table I. Each pattern has a *Name*, a *Before* state, and an *After* state (which is basically an extension of the Before state). The phenomena whose names are shown in braces are placeholders for any phenomenon that conforms to the topology of the pattern. As seen in Table I, these placeholders are used for naming new phenomena in the After state.

A set of code generation patterns, which can be used in order to generate the code of our target system, are illustrated in Table II. Each generation pattern has a Name, an Ivy pattern that corresponds to the generator, and a generation logic that generates or modifies the code of some phenomena using the code of lower-level phenomena. Table III shows a more complex code generation pattern corresponding to class reification (extraction of a class from an association relationship), and Table IV shows an example of its application to a concrete Ivy model.

VII. CONCLUSION AND FUTURE WORK

We have proposed EBGSD as a GSD approach, and have proposed a methodology for applying it to real-world projects. Achieving high levels of reuse, maintainability, and complexity management are some important potential benefits of our proposed approach. Since Ivy generation patterns and code generation patterns are fine-grained patterns expressed at a high level of abstraction, EBGSD can achieve high levels of reuse. EBGSD can increase maintainability and manage software complexity from two aspects: increasing software understandability, and improving modifiability. The Ivy model can be seen as a software construction map, which can be easily reviewed through a graph traversal algorithm. On the other hand, since the couplings among Ivy elements are simple and unidirectional, it is easy to apply the necessary changes, as the changes do not propagate in an unmanageable fashion. Furthermore, because this approach performs all the steps of software development and produces the target artifacts in a smooth and seamless manner, it can be a potential solution to the conflict between modeling and agile development; model-phobic agile methodologies might find it worthwhile to invest in Ivy modeling, as Ivy models are simple and straightforward, and can be used in such a way that agility is not adversely affected.

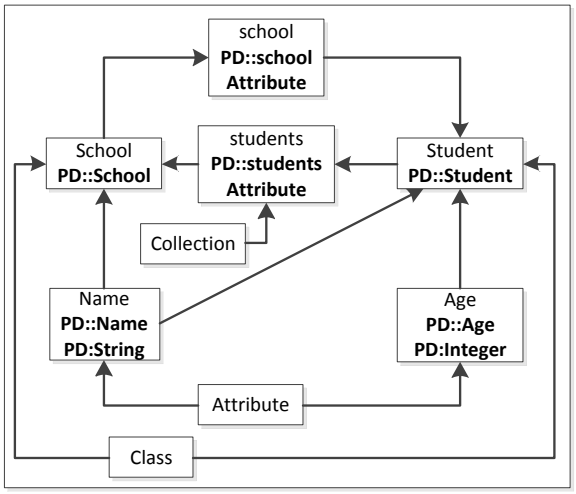


Figure 5. Partial software-solution Ivy model for the education system.

At present, we are evaluating the approach through a case study. Future research can focus on exploring existing opportunities for using the approach for enhancing automation through *construction*, rather than just *reuse*.

REFERENCES

[1] T. Mens, "On the Complexity of Software Systems," Computer, vol. 45, Aug. 2012, pp. 79–81, doi: 10.1109/MC.2012.273.

[2] K. Czarnecki, "Generative Programming," PhD thesis, Technical University of Ilmenau, 1999.

[3] K. Czarnecki, "Overview of Generative Software Development," Proc. European Commision and US National Science Foundation Strategic Research Workshop on Unconventional Programming Paradigms, 2005, pp. 326–341, doi: 10.1007/11527800_25.

[4] R. Poli, W. B. Langdon, and N. F. McPhee, A Field Guide to Genetic Programming, Lulu Enterprises, 2008.

[5] M. Volter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2013.

[6] B. Hailpern and P. Tarr, "Model-driven development: The good, the bad, and the ugly," IBM Systems Journal, vol. 45, July. 2006, pp. 451–461, doi: 10.1147/sj.453.0451.

[7] S. Apel, D. Batory, C. Kastner, and G. Saake, Feature-Oriented Software Product Lines: Concepts and Implementation, Springer, 2013.

[8] J. Royer and H. Arboleda, Model-Driven and Software Product Line Engineering, Wiley, 2012.

[9] M. Besharati, "Generativity in Software Development: Survey and Analysis," M.Sc. Seminar Report, Sharif University of Technology, 2013 (In Persian).

[10] A. Van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specifications, Wiley, 2009.

[11] C. Kastner and S. Apel, "Feature-Oriented Software Development: A Short Tutorial on Feature-Oriented Programming, Virtual Separation of Concerns, and Variability-Aware Analysis," in Generative and Transformational Techniques in Software Engineering IV, J.M. Fernandes, R. Lammel, J. Visser, J. Saraiva, Eds. Springer, 2013, pp. 346–382, doi: 10.1007/978-3-642-18023-1.

[12] P. Schobbens, P. Heymans, and J. Trigaux "Feature Diagrams: A Survey and a Formal Semantics," Proc. International Conference on Requirements Engineering, 2006, pp. 139–148, doi: 10.1109/RE.2006.23.

[13] J. Goldstein , "Emergence in complex systems," in The SAGE Handbook of Complexity and Management, P. Allen, S. Maguire, B. Mckelvey, Eds. SAGE, 2011, pp. 65–78.

 ISBN: 978-1-61208-438-1

TABLE I. A SET OF IVY GENERATION PATTERNS ELICITED FROM THE EXAMPLE

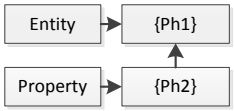
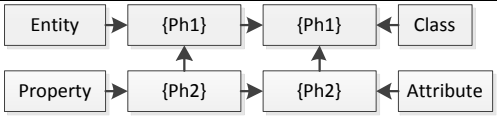
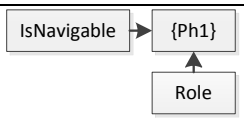
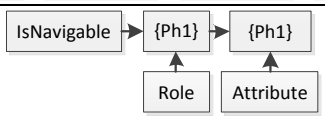
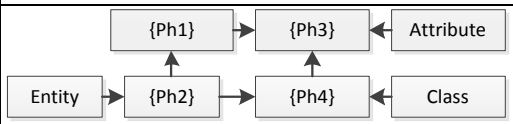
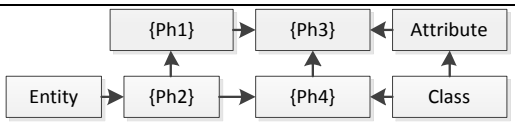
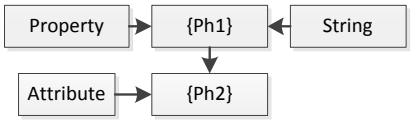
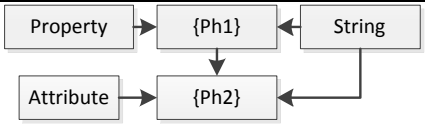
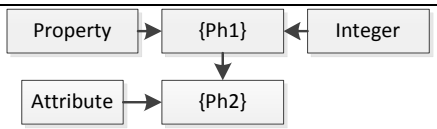
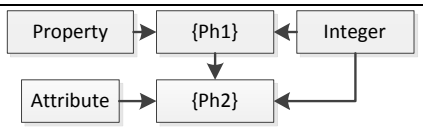
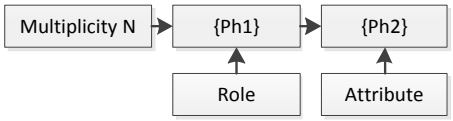
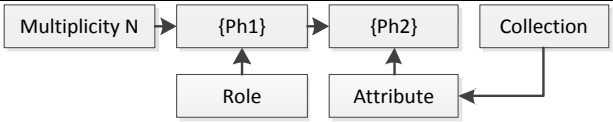
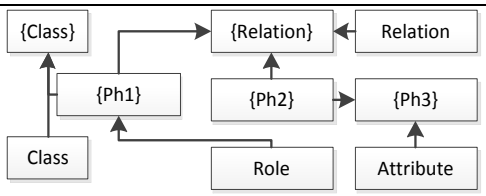
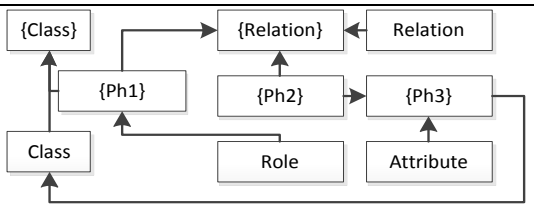
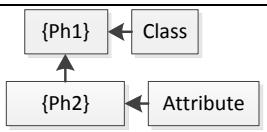
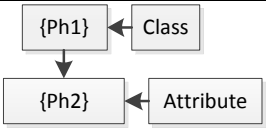
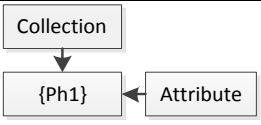
NAME	BEFORE	AFTER
Emergence of Classes and Attributes		
Emergence of Relational Attribute		
Specifying Relational Attribute Type		
Injecting String Type		
Injecting Integer Type		
Emergence of Collections		
Injecting Relational Attribute into Class		

TABLE II. A SET OF CODE GENERATION PATTERNS ELICITED FROM THE EXAMPLE

NAME	PATTERN	GENERATION LOGIC
Class Code Generation		class {Ph1} { ClassTemplate c; Code() { if(c == null) c = Class::Code(); c.SetName({Ph1}); return c; } }
Injecting Attribute Code into Class Code		class {Ph1} { ClassTemplate c; Code() { if(c == null) c = Class::Code(); AttributeTemplate a = {Ph2}::Code(); c.AddAttribute(a) } }
Injecting Type into Attribute Code		class {Ph2} { AttributeTemplate a; Code() { if(a == null) a = Attribute::Code(); a.SetType({Ph1}); return a; } }
Injecting Collection Semantic into Attribute Code		class {Ph1} { AttribueTemplate a; Code() { if(a == null) a = Attribute::Code(); a.SetAsCollection(); return a; } }

 ISBN: 978-1-61208-438-1

TABLE III. A MORE COMPLEX CODE GENERATION PATTERN, CORRESPONDING TO CLASS REIFICATION

NAME	PATTERN	GENERATION LOGIC

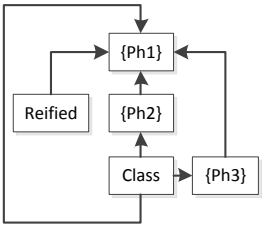
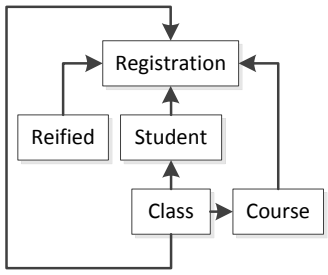
Reified-Class Code Generation		class {Ph1} {
		ClassTemplate c;
		Code() {
		c.SetName({Ph1});
		AttributeTemplate a1 = new AttributeTemplate();
		a1.SetType({Ph2});
		a2.SetName(lowercase({Ph2}));
		c.AddAttribute(a1);
		AttributeTemplate a2 = new AttributeTemplate();
		a2.SetType({Ph3});
		a2.SetName(lowercase({Ph3}));
		c.AddAttribute(a2);
		AttributeTemplate a = new AttributeTemplate();
		a.SetType(c);
		a.SetName(plural(lowercase({Ph1})));
		a.SetAsCollection();
		a.SetAsStatic();
		c.AddAtribute(a);
		MethodTemplate m = new MethodTemplate();
		m.SetName("Add"+{Ph1});
		m.AddParameter(1, {Ph2}, lowercase({Ph3}));
		m.AddParameter(2, {Ph3}, lowercase({Ph3}));
		CreateTemplate cr =
		new CreateTemplate({Ph1}, m.GetParameter(1), m.GetParameter(2));
		AddToCollectionTemplate ad =
		new AddToCollectionTemplate(cr.GetResult(), a);
		m.AddStatement(1, (StatementTemplate)cr);
		m.AddStatement(2, (StatementTemplate)ad);
		c.AddMethod(m);
		return c;
		}
		}

TABLE IV. EXAMPLE OF APPLYING THE CODE GENERATION PATTERN SHOWN IN TABLE III (CLASS REIFICATION)

CONCRETE IVY	TARGET CODE

	class Registration {
	Student student;
	Course course;
	Registration(Student student, Course course) {
	this.student = student;
	this.course = course;
	}
	static Collection<Registration> registrations;
	static AddRegistration(Student student, Course course) {
	Registration registration = Registration(student, Course);
	registrations.Add(registration);
	}
	}

 ISBN: 978-1-61208-438-1

A GPU-aware Component Model Extension for Heterogeneous Embedded Systems

Gabriel Campeanu, Jan Carlson and Séverine Sentilles

Mälardalen Real-Time Research Center
Mälardalen University, Sweden
Email: {gabriel.campeanu, jan.carlson, severine.sentilles}@mdh.se

***Abstract*—One way for modern embedded systems to tackle the demand for more complex functionality requiring more computational power is to take advantage of heterogeneous hardware. These hardware platforms are constructed from the combination of different processing units including both traditional CPUs and for example Graphical Processing Units (GPUs). However, there is a lack of efficient approaches supporting software development for such systems. In particular, modern software development approaches, such as component-based development, do not provide sufficient support for heterogeneous hardware platforms. This paper presents a component model extension, which defines specific features for components with GPU capabilities. The benefits of the proposed solution include an increased system performance by accelerating the communication between GPU-aware components and the possibility to control the distribution of GPU computation resources at system level.**

Keywords–Embedded Systems; Component-based Development; Heterogeneous CPU-GPU Systems; GPU Component Model.

I. INTRODUCTION

In the last years, various embedded computing technologies have emerged due to the rapid advance of microprocessing technology. Homogeneous single-core CPU systems have evolved into heterogeneous systems with different processing units such as multi-core CPUs or GPUs. Taking benefits of the increased computational parallel power, new applications have emerged while others improved their performance. Examples of systems that now use GPU processing hardware include vehicle vision systems [1] and autonomous vision-based robots [2]. However, a GPU is a different hardware unit that has its own memory system. As a result, combining a GPU with a CPU leads to an increase of the software complexity and the need to optimize the use of the available resources.

One way of addressing the increasing system complexity is through component-based development (CBD). In CBD, complex software applications are built by composing already existing software solutions (i.e., software components), resulting in increased productivity, better quality and a faster time-to-market. The approach has been successfully used in other domains, but has recently attracted attention also for developing software for embedded systems, as evident by industrially adopted component models such as Autosar [3] and Rubus [4]. In order to address the specifics of embedded systems, many component models targeting this domain follow a *pipe & filter* architectural style. Using this style, the components are passive and the transfer of data and control is defined statically by how they are connected, rather than the typical object oriented style with active components and method calls [5].

Having no component model support for GPU development, each component that needs to use the GPU must encapsulate various GPU specific operations such as memory initialization, and operations to shift data between the CPU and GPU memory systems. This introduces a communication overhead among components (i.e., leading to longer response times) and unnecessary code duplication. Also, these components have to encapsulate all the GPU settings required to meet their functionality. For example, each component independently decides how much GPU computation resources it uses (e.g., number of threads), which can result in a suboptimal GPU usage in the system as a whole, decreasing the system performance. The lack of efficient development methods affects several component properties (e.g., granularity, reusability), making difficult to fully use the benefits of GPU systems.

In this paper, we describe how the problem can be tackled by proposing a GPU-aware component model extension where components are equipped with GPU ports that allow component communication directly through the GPU environment. We also provide a way for the system to decide the GPU settings (e.g., number of threads) for each GPU-aware component. Enhancing the communication and delegating the component GPU settings to the system level, result in increased system performance while keeping the key benefits of the CBD approach.

The rest of the paper is organized as follows. Section II gives a background depiction of existing component model challenges in addressing efficiently the GPU hardware. A high level descriptions of the GPU-aware components is covered by Section III, where the specification of the component interfaces and GPU ports are described in depth. Section IV describes a running example which illustrates the underlying details of our solution. In Section V, a series of experiments were carried out to evaluate the performance efficiency of the proposed method. Related work is described by Section VI, while Section VII presents the paper conclusion and future work.

II. USING GPUS IN COMPONENT-BASED DEVELOPMENT

When developing applications for heterogeneous embedded systems using a *pipe & filter*-type of component model, the developer follows the model specifications to develop software components. The same specifications are used even when the model does not provide directions on how to support GPU within the component. Hence, a component with GPU computations requires encapsulation of all the information and operations needed to support its functionality. For example, the information and operations include choosing the GPU computational resources (e.g., number of threads, grid dimension) to process data or, being GPU unaware, operations to replicate data from and to the main (CPU) memory system.

 ISBN: 978-1-61208-438-1

In the following example, we present a part of a component-based software architecture of a demonstrator that uses a heterogeneous CPU-GPU hardware platform. The demonstrator is an underwater robot developed at Mälardalen University, Sweden. It is used as the running case for the RALF3 research project [6]. The architecture is an abstract view of a component model design that uses a *pipe&filter* interaction style (e.g., ProCom [7], Rubus [4]). The robot has an embedded electronic board containing a GPU, alongside the common CPU. Both processing units have different memory systems. The board is connected to the cameras that are providing a continuous stream of images, and to other sensors and actuators (e.g., pressure sensors, motors).

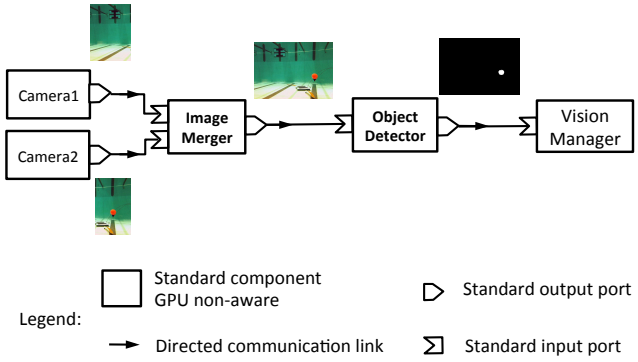


Figure 1. The abstract software architecture of the vision system

Figure 1 presents the robot vision system architecture combined with the data propagation view. The robot uses two cameras which give an extended perspective of the surrounding underwater environment. The physical cameras provide a continuous stream of frames to *Camera1* and *Camera2* software components. The *ImageMerger* receives the frames from the camera components and merge them into a single frame that is filtered by *ObjectDetector*. The filtering process produces a black-and-white frame which eases the identification process of specific objects (e.g., red buoys). The vision system uses the parallel processing power of the GPU hardware for its main data processing activities from *ImageMerger* and *ObjectDetector* components. The black-and-white frame is received by the *VisionManager* component which, based on the position of the objects that have been detected, takes movement decisions for the underwater robot.

Each component, as part of its functionality, accesses particular hardware elements (e.g., CPU, RAM, GPU) in a specific order. The hardware related activities of the vision system are illustrated in Figure 2. The *Camera1* and *Camera2* components, connected to the physical cameras, fetch data frames (two at a time) onto the main (CPU) memory system. The *ImageMerger* component duplicates the two frames onto the GPU memory system and processes them (i.e., merge them into one frame). The component handles inside various operations such as memory allocation, specific GPU-shifting operations and picking suitable GPU computational settings for its processing activity. Having GPU-unaware communication ports, the connection with the *ObjectDetector* component is done in a form that is recognized by the existing component interfaces, i.e., thought the main system. Using the same component model rules, the *ObjectDetector* component has similar actions.

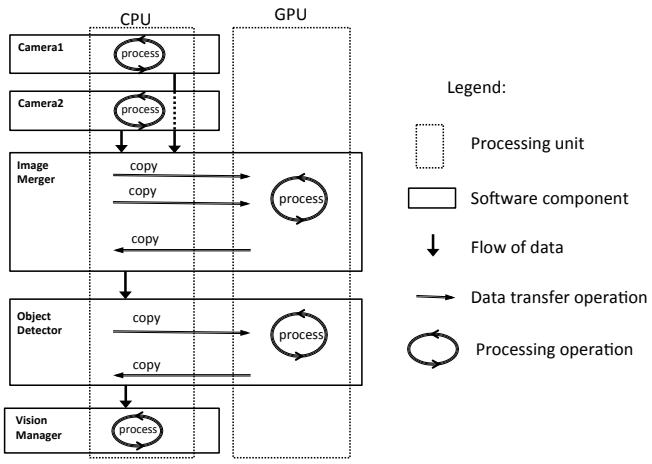


Figure 2. Vision system activities over the hardware

In general, using a *pipe & filter*-type of component model to develop applications for heterogeneous embedded systems has the following disadvantages:

- By being responsible for transferring the data between processing units, each component with GPU capability uses an inefficient copying mechanism as communication method. In most cases, this results in an increased communication over the CPU-GPU hardware bridge (e.g., PCI-Express), which decreases the system performance (e.g., worse reaction time).
- As a side effect of the component encapsulating the same transfer operations, the system contains duplicate code. That is each GPU-based component has copy-from or copy-to CPU operations.
- Each component with GPU capability individually decides (at the development phase) the computational configuration settings. This affects the overall GPU usage of the system and also makes the component less reusable in other contexts.

III. THE GPU-AWARE COMPONENT MODEL EXTENSION

To overcome the drawbacks of using *pipe & filter* component models with no GPU support, we propose a GPU-aware component model extension. In summary, the solution introduces:

- A standardized *configuration interface* through which a component receives GPU computational settings. The assigned settings (limited by the hardware constraints) have a direct impact on the performance of the application. The system, knowing the underlying hardware platform, takes the decision of the computational resources distribution among GPU-aware components. For example, it may distribute the GPU computational resources in such a way that several components can run in parallel (e.g., their summed number of threads should not exceed the total GPU number of threads).
- Dedicated *GPU ports* which are aware of the GPU environment. Instead of communicating using the main

 ISBN: 978-1-61208-438-1

memory, the GPU-aware components communicate directly using the GPU memory.

- Automatically generated *adapters* with dedicated transfer operations. The adapters are automatically introduced when a GPU port is connected to a standard port, in order to facilitate the data transfer operation between the processing units.

According to our proposed solution, the architectural software model of a heterogeneous embedded system is extended in the following way. Ports are classified as GPU ports or standard ports, and the model contains two types of components, GPU-aware and standard components. Standard components can only have standard ports while the GPU-aware components can have both GPU and standard ports. In addition, the software model contains two new inter-component communication elements, the automatically generated communication adapters (CPU-to-GPU and GPU-to-CPU) that resolve the data incompatibility issue between the port types.

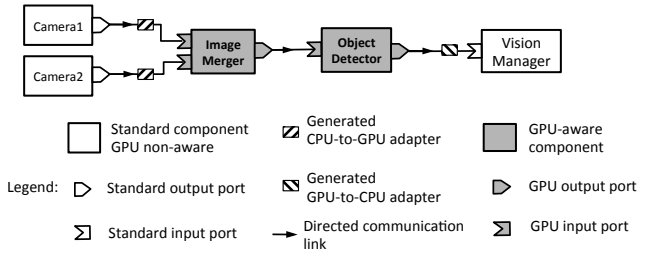


Figure 3. The abstract software architecture of the vision system using a GPU-aware solution

Figure 3 illustrates the abstract software architecture of the vision system using our proposed solution. The system has two GPU-aware components, i.e., *ImageMerger* and *ObjectDetector*, that uses their GPU ports to communicate via the GPU environment. The data provided by the *Camera1* and *Camera2* components are placed onto the GPU by automatically generated CPU-to-GPU adapters. After the GPU-aware components finish their functionality, the output is placed on the main (CPU) memory by another generated adapter (i.e., GPU-to-CPU adapter). This makes the output of the *ObjectDetector* available to the *VisionManager* component.

At startup, the system communicates in a transparent way with the GPU-aware components, connecting to their standardized configuration interfaces. From the architectural software perspective, however, this system-to-component communication mechanism is not graphically represented.

Figure 4 presents the vision system activities of the GPU-adapted software architecture. Compared to the previous non-GPU-aware solution from Figure 2, the newly introduced adapters are handling the data transfer between the CPU and GPU. The two first adapters move data onto the GPU, while the third one transfers the final result back onto the CPU. The *ImageMerger* component, using its GPU ports, takes the frames directly from the GPU (where the adapters placed them) and processes them using the hardware configuration setting received from the system. Also, by having dedicated GPU ports, the communication with *ObjectDetector* is done locally via the GPU memory.

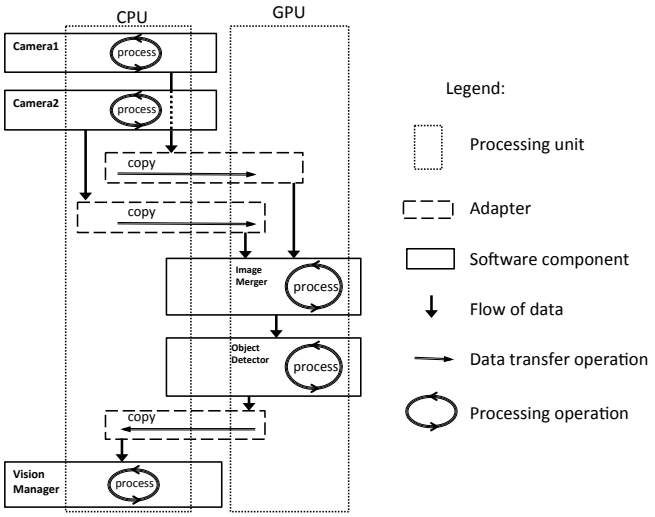


Figure 4. GPU-aware vision system activities over the hardware

The main advantages of our GPU-aware solution are the following:

- Keeping the component communication local on the GPU, whenever possible. This improves the system performance (e.g., better component-to-component communication time) and decreases the communication stress over the hardware CPU-GPU bridge.
- By externalizing the data shifting operations from the component, the component granularity is improved. Also, as a consequence of introducing the specialized adapters, duplicated code is reduced (when the system has at least two GPU-aware components sequentially connected).
- Deciding the GPU configuration of each component at the system level improves the reusability of components. For example, the system can run several components in parallel on the GPU by adjusting their GPU configuration settings, or the same component can be used in different systems with different configuration settings.

IV. EXTENSION IMPLEMENTATION

Next, we give an example of what the extension might look like when implemented. We base the presentation on a simple reference component model to simplify the presentation, and use the vision system from Figure 3 as a running example.

A. An implementation of GPU-aware components

Using C++ as the implementation language, we see a software component as an object with its public member functions describing the component ports and interfaces. Any GPU-aware component is characterized by four member functions, as presented in Figure 5.

The first argument set of the *initialize* function, (i.e., *SIZE_TYPE frame1_in, ...*) specifies the data sizes of the input ports, while the second set (i.e., *SIZE_TYPE frame1_out, ...)* describes the data sizes of the output ports. For GPU-aware components, the *initialize* function has the role of allocating memory on the GPU to hold the produced output

 ISBN: 978-1-61208-438-1

```
class GPU_awareComp{
public:
  void initialize(SIZE_TYPE frame1_in, ...,
                  SIZE_TYPE *frame1_out, ...);
  void config_gpu(int tile, int block_x, int block_y, int block_z);
  void execute(GPU_TYPE in1, ..., GPU_TYPE *out1, ...);
  void free_memory();
};
```

Figure 5. A GPU-aware component implementation details

data. Based on the sizes of the input data, the component uses a GPU API routine (e.g., cudaMalloc) to allocate a specific chunk of memory. The sizes of the allocated memory (i.e., *SIZE_TYPE frame1_out, ...*) will be propagated to the next connected component as input data sizes. The (input and output) arguments are of a structure type and may hold several elements indicating the multi-dimensional aspect of the data, such as three-dimensional matrices or bi-dimensional images. For example, a 2D image argument may be represented by a data structure with two member elements (width and height) that specify the number of image pixels.

Each GPU-aware component receives from the system its GPU execution settings, through the *config_gpu* function. The function parameters describe this settings, such as the three-dimensional size ($block\_x * block\_y * block\_z$) of the thread-blocks unit. A thread-block is a specific unit of thread organization.

The *initialize* and *config_gpu* functions are used once at system startup. After a component allocates the memory to hold its results, it reuses it for all of its executions. The same principle applies for the GPU execution setting; once a component has the GPU setting, it is used every time the component is executed. This design decision is suitable for relatively simple stream processing applications, with static control flow and where one frame is fully processed before starting on the next. Each time a component is invoked, it processes different data using the same GPU setting and reusing the same allocated memory space (considering the dimensions of the streaming frames do not change over time), avoiding the overhead of allocating/deallocating and specifying the execution setting for each frame of the stream.

The *execute* function triggers the core functionality of a component. This function is specified with two sets of arguments. The first set (e.g., *GPU_TYPE in1, ...*), corresponding to its input ports, are pointer variables that specify the GPU locations of the input data. The second set (i.e., *GPU_TYPE *out1, ...*) indicates the GPU locations of the component results, corresponding to the output ports. In case when the component also has standard ports, the types of the ports are used (e.g., TYPE instead of GPU_TYPE).

After the component finishes its executions (it may run several times), it must free the memory that has been allocated to hold its output results. This is done by the *free_memory* function, which uses a GPU API deallocation routine (e.g., cudaFree).

B. Adapters implementation

The GPU-aware components, by communicating directly via the GPU memory, do not have to handle the data shifting activities. Instead, we propose automatically generated software adapters to handle the data transfer between the two computational units.

```
class Cpu2Gpu_adapter{
public:
  void initialize(SIZE_TYPE frame_in);
  void transfer(TYPE in, GPU_TYPE *out);
  void free_memory();
};
```

Figure 6. A CPU-to-GPU adapter implementation details

Figure 6 describes the interface of an adapter that handles the CPU-to-GPU data transfer. Through the *initialize* function, the adapter receives from the system the size of the data to be transferred. Based on this, it allocates memory space on the GPU using a GPU API procedure. The parameter of the *initialize* function, being of a structure type, may hold several elements, which reflects the multi-dimension aspect of the *frame_in*. The *transfer* function uses a GPU API copy procedure (e.g., cudaMemcpy) to transfer data. The first argument represents the CPU location of the input data (from the main memory system), while the second argument holds the GPU location where the data was transferred. The *free_memory* interface deallocates the memory space that was allocated on the GPU.

```
GPU_UCHAR *dev_ptr;

void initialize(SIZE_TYPE frame_in) {
  cudaMalloc(&dev_ptr, 3 * sizeof(dev_ptr) * frame_in.width *
      frame_in.height);
}

void transfer(unsigned char *in, GPU_UCHAR **out) {
  cudaMemcpy(dev_rgb, host_ptr, 3 * sizeof(in) * frame_in.
      width * frame_in.height, cudaMemcpyHostToDevice);
  *out = dev_ptr;
}

void free_memory() {
  cudaFree(dev_ptr);
}
```

Figure 7. An example of CPU-to-GPU adapter implementation using CUDA API

Figure 7 illustrates the implementation details of a CPU-to-GPU adapter using the CUDA API programming model. The adapter transfers a bi-dimensional RGB image (red, green and blue elements of a frame pixel) from the main (CPU) memory to the GPU memory. The input image argument (*img*) is represented by a *SIZE_TYPE* data structure that contains two member elements, *width* and *height*. The *transfer* function uses the main memory frame location specified by the input argument *in*, and executing the *cudaMemcpy* routine, places the image onto the GPU. The memory location of the newly shifted image is memorized by the output argument *out*. In order to make a distinction between the two different memory systems (CPU and GPU), we use two different types to characterize the input and output arguments, i.e., the *unsigned char* and

```
struct SIZE_TYPE {
        int height;
        int width;
};

SIZE_TYPE frame1_in, frame2_in, frame_mrg, frame_filtered;
unsigned char *camera1_in, *camera2_in, *result;
GPU_UCHAR *adp1, *adp2, *merge, *obj;

Camera1.initialize(&frame1_in);
Camera2.initialize(&frame2_in);
Adapter1_CPU2GPU.initialize(frame1_in);
Adapter2_CPU2GPU.initialize(frame2_in);
ImageMerger.initialize(frame1_in, frame2_in, &frame_mrg);
ObjectDetection.initialize(frame_mrg, &frame_filtered);
Adapter3_GPU2CPU.initialize(frame_filtered);
VisionManager.initialize(frame_filtered);

ImageMerger.config_gpu(16, 16, 16, 1);
ObjectDetection.config_gpu(32, 16, 16, 1);

while(stream!=NULL) {
  Camera1.execute(&camera1_in);
  Camera2.execute(&camera2_in);
  Adapter1_CPU2GPU.transfer(camera1_in, &adp1);
  Adapter2_CPU2GPU.transfer(camera2_in, &adp2);
  ImageMerger.execute(adp1, adp2, &merge);
  ObjectDetection.execute(merge, &obj);
  Adapter3_GPU2CPU.transfer(obj, &result);
  VisionManager.execute(result);
}
```

Figure 8. The implementation details of the vision system

GPU_UCHAR types for the main memory and GPU memory location, respectively.

The GPU-to-CPU adapter is implemented in a similar way. The only major difference is located in the *transfer* function, where the first argument describes the GPU location of the image to be transferred, while the other argument describes the main memory (CPU) location of the transferred data.

C. Vision system implementation

We now use the GPU-aware component and adapter implementations previously described to present our implementation of the vision system.

For the robot's vision system, three adapters are automatically generated: two for placing images on the GPU and the other for shifting the final result back on the main (CPU) memory. The initialization of the adapters and GPU-aware components are specified in the upper part of the Figure 8.

The image sizes of the camera output are propagated to the rest of the system according to each component's functionality. For example, the *ImageMerger* component receives the input sizes of the two camera images, and outputs the size of the merged image to the next connected component (*ObjectDetection*). The initialization part is done only once, each adapter and GPU-aware component reusing the same allocated memory to place the continuous stream of frames received from the cameras.

The GPU setting of each GPU-aware component is provided by the system through the *conf_gpu* methods. The system sends once the execution setting to each of the components, which is reused for every image processing activity of the components during the entire application execution. For example, the processing unit of *ObjectDetection* is a block of $16 * 16 * 1$ threads, while is applied on frame tiles of $32 * 32$ pixels.

The execution of the system core functionality is illustrated in the bottom part of the figure, inside the *while* loop. As long as the stream frame flow is not closed (it stops when e.g., the robot mission is completed), the camera components are producing frames which are copied onto the GPU by the CPU-to-GPU adapters. *ImageMerger* uses its input parameter pointers that indicate the GPU memory location of the frames, and outputs, using a GPU-type pointer variable, the location of its result. In the end, the *VisionManager*, using the memory location provided by the GPU-to-CPU shifted data, it processes the data, taking appropriate movement decisions of the underwater robot.

V. Evaluation

To examine the benefits of our proposed solution, we conducted a small experiment to compare the performance with and without the GPU-aware extension, to determine the reduction in communication overhead. To keep it simple, we use only one component, i.e., vertical mirroring of an image, implemented in two variants. A GPU-aware component, developed using our solution, and a standard component developed as described in Section II (encapsulating the data shifting operations between CPU and GPU). We then construct systems of difference sizes by connecting multiple (from 5 to 25) component instances sequentially, using either the GPU-aware or the standard variant.

Two input images are used, one with $1152 * 864$ pixels and a second, larger, with $1152 * 1782$ pixels. The platform on which the experiments were executed consists of an NVIDIA GPU hardware with a Kepler architecture, a 2,6 GHz IntelCore i7 CPU with 16 GB of internal RAM memory. For each case, we executed the system 100 times and calculated the average of the measured times.

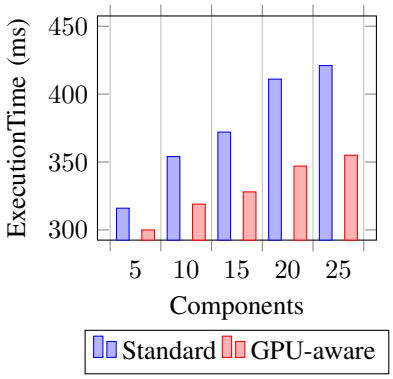


Figure 9. Execution times of two types of systems when processing an image with $1152 * 864$ pixels

Figure 9 illustrates the execution times of the two types of systems while processing an input image of $1152 * 864$ pixels. With standard components, represented in blue color, it takes approximately 315 ms for 5 component to sequentially process

the input image, and 420 ms when the system consists of 25 components. The GPU-aware variant, depicted with red color, need approximately 300 ms to process the same image with 5 components, and 355 ms when the system is composed of 25 components.

The results for the larger input image of $1152*1782$ pixels, presented in Figure 10, show similar improvements.

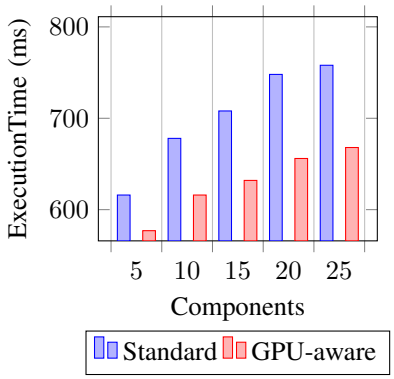


Figure 10. Execution times of two types of systems when processing an image with $1152 * 1782$ pixels

The experiment shows a performance increase of the GPU-aware solution over the standard one where GPU interaction is completely encapsulated in the components. For the smaller input data, the gain is approximately 5% of the total execution time with 5 components, and 16% with 25 components. For the larger input, the gain is 6% and 12% with 5 and 25 components, respectively.

VI. Related Work

There are several component-based approaches that in different ways target GPU-based systems, as discussed in the following paragraphs.

Elastic Computing [8] is a framework that, based on a library that contains pre-built "elastic functions" for specific computations, determines offline the optimized execution configuration for a given platform. The framework does not manage the execution of GPU devices, which is done internally inside the elastic functions, alongside with the resource allocation and data management.

Kicherer et al. [9] use a component-based approach to propose a performance model suitable for on-line learning systems. The disadvantage of their approach is that the data management does not handle transfer operations for the GPU execution. Hence, the data transfer between the main memory and the GPU device is done internally by the library of the performance model. Differing from both of the presented approaches, the data management and resource allocation is done automatically (by adapters), from outside of the component level.

A theoretical component model is proposed by Stoinski [10] to support data stream applications by adding a dedicated port which enables data stream communication (e.g., MPEG1 video) between components. Comparable to this theoretical approach, we are extending the hardware platform specification to include GPUs, and enriching the component interfaces to enable data (e.g., stream of frames) communication between components via the GPU memory system.

The PEPPHER component model [11] constructs an environment for annotations of C/C++ components for heterogeneous systems, including (multi-)GPU based systems. The model provides different (sequential or parallel) implementation variants (e.g., one for multi-core CPU and another for GPU) for the same computational functionality (component), together with the meta-data (tunable parameters). The composition code of the component is in the form of stubs (proxy or wrapper functions). In addition to this work, we address, transparently, the system-to-component communication for the GPU execution settings. The memory management issue is handled by smart containers. Contrasting their approach, we use automatically generated adapters which can be seen as a high level memory management elements.

Regarding code generation, there is much work done in automatically porting sequential (or parallel) CPU source code for GPU execution. Several programming languages have such GPU translators, such as Java [12], C [13], C++ [14], OpenMP [15], Python [16] or Matlab [17]. For our work, these approaches can be used to generate parts of the implementation of GPU-aware components.

VII. Conclusion

Despite the growing trend of using heterogeneous platforms for embedded systems, there is a lack of efficient ways to address the CPU-GPU combination in the existing *pipe & filter* component models. When a component model does not provide dedicated means to specifically handle GPUs, each component have to redundantly encapsulate the same GPU specific operations and settings required to meet and support their functionality. Our solution tackles the inefficient development by proposing a GPU-aware component model extension. With our method, the components are aware of the GPU environment by having specialized GPU interfaces and ports which facilitates the component communication via the GPU environment.

The benefits of our solution include:

- The system performance is increased from reducing the component communication overhead and keeping data locally on the GPU when possible, as indicated by the experiment in Section V.
- Improved component granularity and reduced code duplication, as a consequence of introducing specialized generated adapters for data shifting operations.
- An increased reusability of components by adjusting the components GPU configuration setting at the system level. For example, the system can run several components in parallel on the GPU by adjusting their GPU configuration settings, or the same component can be used in different systems with different settings.

As future work we want to increase the flexibility of the GPU memory management to support also more dynamic memory allocation. Moreover, our work may be extended by supporting parallel execution of GPU-aware components. Another possible thread of future work includes implementing the method in some existing component model, e.g., Rubus [4] or ProCom [7].

 ISBN: 978-1-61208-438-1

ACKNOWLEDGMENT

Our research is supported by the RALF3 project [18] through the Swedish Foundation for Strategic Research (SSF).

REFERENCES

[1] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 7, 2010, pp. 1239–1258.

[2] P. Michel et al., "GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing," in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE, 2007, pp. 463–469.

[3] AUTOSAR Development Partnership, "AUTOSAR Technical Overview, v4.2," http://www.autosar.org, (accessed June 28, 2015).

[4] Arcticus Systems, "Rubus Component Model," https://www.arcticus-systems.com, (accessed June 28, 2015).

[5] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. Chaudron, "A classification framework for software component models," IEEE Transaction of Software Engineering, vol. 37, no. 5, October 2011, pp. 593–615.

[6] C. Ahlberg et al., "The Black Pearl: An autonomous underwater vehicle," Mälardalen University, Tech. Rep., June 2013, published as part of the AUVSI Foundation and ONR's 16th International RoboSub Competition, San Diego, CA.

[7] S. Sentilles, A. Vulgarakis, T. Bureš, J. Carlson, and I. Crnkovic, "A component model for control-intensive distributed embedded systems," in Proceedings of the 11th International Symposium on Component Based Software Engineering (CBSE2008). Springer Berlin, October 2008, pp. 310–317.

[8] J. R. Wernsing and G. Stitt, "Elastic computing: A portable optimization framework for hybrid computers," Parallel Computing, vol. 38, no. 8, 2012, pp. 438–464.

[9] M. Kicherer, F. Nowak, R. Buchty, and W. Karl, "Seamlessly portable applications: Managing the diversity of modern heterogeneous systems," ACM Transactions on Architecture and Code Optimization (TACO), vol. 8, no. 4, 2012, p. 42.

[10] F. Stoinski, "Towards a component model for universal data streams," Eighth IEEE International Symposium on Computers and Communication, 2003, 2003.

[11] U. Dastgeer, L. Li, and C. Kessler, "The PEPPHER composition tool: Performance-aware dynamic composition of applications for GPU-based systems," in High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:. IEEE, 2012, pp. 711–720.

[12] Y. Yan, M. Grossman, and V. Sarkar, "JCUDA: A programmer-friendly interface for accelerating Java programs with CUDA," in Euro-Par 2009 Parallel Processing. Springer, 2009, pp. 887–899.

[13] M. M. Baskaran, J. Ramanujam, and P. Sadayappan, "Automatic C-to-CUDA code generation for affine programs," in Compiler Construction. Springer, 2010, pp. 244–263.

[14] F. Jacob, J. Gray, Y. Sun, and P. Bangalore, "A platform-independent tool for modeling parallel programs," in Proceedings of the 49th Annual Southeast Regional Conference. ACM, 2011, pp. 138–143.

[15] S. Lee, S.-J. Min, and R. Eigenmann, "OpenMP to GPGPU: a compiler framework for automatic translation and optimization," ACM Sigplan Notices, vol. 44, no. 4, 2009, pp. 101–110.

[16] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation," Parallel Computing, vol. 38, no. 3, 2012, pp. 157–174.

[17] A. R. Brodtkorb, "The graphics processor as a mathematical coprocessor in MATLAB," in Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on. IEEE, 2008, pp. 822–827.

[18] RALF3, "Software for Embedded High Performance Architecture," http://www.mrtc.mdh.se/projects/ralf3/, (accessed September 10, 2015).

 ISBN: 978-1-61208-438-1

Soft System Stakeholder Analysis Methodology

Markus Kelanti, Jarkko Hyysalo, Samuli Saukkonen,
Pasi Kuvaja, Markku Oivo
Department of Information Processing Science
University of Oulu
Oulu, Finland
{markus.kelanti; jarkko.hyysalo; samuli.saukkonen;
pasi.kuvaja; markku.oivo}@oulu.fi

Jari Lehto
Nokia, Networks
Espoo, Finland
jari.lehto@nokia.com

***Abstract*— Understanding problems and the values of any solution from multiple stakeholder perspectives is a fundamental feature of stakeholder analysis. As modern systems increase in size and functionality and include services and other non-software or hardware components, more stakeholders are involved. These stakeholders have different interests and needs, which are often expressed in a multitude of ways. Describing problems and identifying the local strategic values in a constantly changing business environment is strategically important to companies. This paper describes action research conducted within a large, global telecommunication company to study how stakeholder analysis can support software-intensive systems development. The results of the study demonstrate the need to analyse, structure and identify problems and solutions with different local and strategical values. Furthermore, the results show the importance of method usability and the role of stakeholder analysis in supporting software intensive systems development. The outcome of the study was a method for a practical stakeholder analysis that supports the identified needs in the software-intensive systems development.**

Keywords-Software Intensive Systems; Stakeholder; Stakeholder Analysis; Action Research.

I. INTRODUCTION

In today's world, software systems development is becoming increasingly challenging. Software systems are typically not developed by a single company. Rather, they are developed globally, with collaboration between subcontractors, third-party suppliers and in-house developers. Modern systems like software intensive systems (SIS) have not remained local isolated applications, but have become large and complex systems with increased communication with other systems and attached services. Few can master the entire process of product development, so several experts pool their expertise and work together, especially when the software is the main component and affects the product's usability, functionality, development tools and methods, production mechanisms and innovation. Such systems are known as SIS [1][2][3].

The case company, Nokia Networks, develops very large-scale SIS for the global telecommunication market. Development occurs across several countries, which requires knowledge workers [4][5] with a common understanding, shared goals, awareness and practices that support the work. Knowledge-intensive processes are characterised by dynamic changes of goals, information, environment and constraints, as well as intensive individual ad hoc communication and collaboration patterns; thus, it is not easy to plan the work in detail beforehand [6]. SIS are often very large-scale systems, and various stakeholders from different organisations work in collaboration, forming teams that are dynamically and spontaneously assembled and work together via communications technology [7].

In the case of Nokia Networks, large-scale SIS developers often find it difficult to approach, collect, analyse and structure all the information that is currently available. Similar challenges are presented by the information that is obtained as time progresses. The company has adapted agile development approaches to answer these problems, and the individual teams have especially benefitted from this approach. When, however, products or features are analysed from the platform or architectural level, the problems still exist. These challenges often result from ill-defined goals and evaluation criteria and require changes in goals and plans during development. Multiple actors and perspectives, incommensurable and/or conflicting interests, important intangibles, and key uncertainties are typical in such situations [8].

To solve problems like this, a common approach in Requirements Engineering (RE) is to perform a *stakeholder analysis* [9][10]. This type of analysis aims to discover the stakeholders relevant to the problem faced by the developers. Analysing the stakeholders' needs allows the real system and environment of the stakeholders to be defined and the problem to be further understood from multiple perspectives. Utilising this information, a company can design an initial solution and negotiate with stakeholders to solve conflicting interests and produce a solution that results in the most value for the stakeholders and the company. However, the environment in which the company works is highly competitive and dynamic, requiring speed and agility from the development process. There are often uncertainties about whether information is valid or common enough when new features or products are developed. In addition, the development situation also changes as more information becomes available. Uncertainty and changes are common in software development because the processes are complicated

 ISBN: 978-1-61208-438-1

and not all circumstances can be predicted [12][13]. Furthermore, the order in which activities are executed is not necessarily important—it may even be impractical as interaction with the environment, activities and underlying business logic determines the order of execution rather than predetermined, static process schema [11]. The analysis of information is often done intangibly, since the development process involves numerous personalities, experiences, types of education and backgrounds. Understanding and processing is essential for properly structuring information as it helps the company identify the real problem, determine where it can potentially gain the most value and discover the type of solution that is capable of realising this value.

Therefore, the research problem of this paper is: how can problems in SIS development be described and structured using stakeholder analysis? In particular, *how can problems in an SIS development environment be described and analysed in order to identify the impact and value from different stakeholder viewpoints?*

Action research [14] was conducted over a period of one year in the case company to research and develop a practical way to perform stakeholder analysis. The rest of the paper is structured as follows. Section 2 presents the related research, and Section 3 presents the research method. Section 4 describes how the action research cycles were performed and their results. Section 5 presents the soft system stakeholder analysis methodology (S3AM) developed as a result of the action research. Section 6 discusses the results of the research and how problems in SIS should be approached and section 7 describes threats to validity and the limitations of the research. Section 8 presents the conclusions and future topics.

II. RELATED RESEARCH

Freeman [15] was the first to popularise the concept of 'stakeholder'. He defined a stakeholder as a group or individual affected by the achievement of an organisation's objectives, or a group or individual that can affect them. This concept introduced ethical thinking to businesses, causing a company to consider other stakeholders' benefits rather than just stockholders' [16]. This is known as a stakeholder approach where company needs to identify stakeholders in order to identify their needs and manage them [17].

Stakeholder analysis is an internal part of RE in any software development process. An RE stakeholder is generally a person, group or organisation that has an interest in or is connected to the system under development [9][10]. Common stakeholders are end users, engineers, managers and customers [18][19]. Stakeholder analysis is generally integrated into the specific RE method and does not exist as an independent method. It mainly supports the [9][10][19]:

- identification of relevant stakeholders,
- elicitation of stakeholder requirements,
- analysis of requirements from stakeholders' perspectives,
- validation of requirements,
- negotiation of requirements with stakeholders, and
- prioritisation of requirement implementation.

A common approach is to utilise user stories, requirement templates or other structured or semi-structured data containers to capture information about a system and how it works from a stakeholder's perspective. This process can be guided by practical perspective [20] or used to support negotiations [21].

The main critique of stakeholder analysis is that it is not systematic and well defined [18][22]. Either it supports few activities in the development process or its instructions and process are vaguely described. To counter this problem, multiple stakeholder analysis methods have been developed.

Ballejos and Montagna [18] describe a specific method for stakeholder identification in an inter-organisational environment utilising generic stakeholder categories. McManus [22] defines a general systematic approach to stakeholder analysis that describes the identification, elicitation, analysis and negotiation processes in RE. McManus also provides an identification and analysis method based on the World Bank's list of possible stakeholders. Alexander and Robertson [33] use the onion model to identify and involve stakeholders to the development process. Lim et al. [23] utilises social networks to systematically analyse stakeholders in large-scale software projects. They utilise crowdsourcing to automate stakeholder analysis by asking the stakeholders to recommend other relevant stakeholders and aggregating the answers via social network analysis.

While there are known and established methods for stakeholder analysis, none has established itself as the benchmark method in a development effort dominated by software. Furthermore, the methods typically concentrate on small- and medium-scale development efforts. However, instead of abolishing any of the existing methods, it is worthwhile to examine how the stakeholder analysis should work in SIS development context and how it should be implemented in order to benefit its users.

III. RESEARCH SETTING AND PROCESS

To monitor their development process, the case company utilises metrics that measure the process on different levels. Data is collected and then synthesised for different stakeholders to compare the development to pre-defined guidelines. This system is known as a metrics reporting system (MRS). The main benefit of this type of system is that it helps analyse and visualise the statuses of different organisational units and the overall picture.

The data for the MRS is provided and calculated by multiple units. The units participating in this research had been experiencing organisational changes: some of the work was done manually, and some was assisted by tools. The company was interested in finding a way to automate the reporting system—or at least parts of it—to try to avoid manual data collection. Its goals were to determine the extent to which the MRS could be automated, discover its requirements and calculate the value of the automation. In this case, part of the MRS (8 metrics) was deployed to measure the performance of product lines. The metrics reported the data for each product line at the end of a reporting period.

 ISBN: 978-1-61208-438-1

In order to reach these goals, the company needed a method for collecting and structuring information to describe the current MRS from multiple stakeholder perspectives. This would allow for the analysis of changes to the system and their potential value. Because the current understanding of the MRS was unclear and there were a number of unknown variables, action research [14] was selected as the research method. It was important to establish constant collaboration between the company, participants and researchers to ensure a complete understanding of the environment and problem. Action research offered flexibility and an iterative approach to the problem.

Action research is an iterative and systematic process that addresses concrete organisational problems through the application of theories in practice. It allows both researchers and practitioners to gradually create a satisfactory solution to the organisation's problem while adding to the scientific knowledge on the topic [24][25]. Action research is composed of cycles or iterations, allowing for constant re-evaluation of the problem, implementation of the solution and learning in short intervals [14][26][27]. The cycles are divided into the following phases:

- diagnosis, where the problem is identified, analysed and defined;
- action planning, where the actions to address the defined problem are decided based on the available solutions and theories;
- action taking, where the desired actions are implemented;
- evaluation, where the impact of the action is studied; and
- specify learning, where the results and findings of the evaluation are documented and published, and then this information is used in a new cycle.

The action research method in this study was implemented as follows. A pre-study was conducted to analyse the company's problem and select a suitable approach from the literature. After the pre-study, the iterations began and were continued until a suitable solution was devised. Each iteration began with a diagnosis meeting between at least two company representatives and one researcher, who analysed the results of the pre-study or a previous iteration. Based on the results, they determined the desired actions for the iteration and how long the iteration would last. After the meeting, the actions were implemented; this was done primarily by the researcher, who was assisted by the company representatives. When this phase ended, the researcher evaluated the results and presented them in a retrospective meeting. The purpose of the retrospective meeting was to specify what was learned. This meeting is open to the company representatives and other company personnel, especially those who were involved in the research.

A total of five iterations were completed during 2014, and a total of 30 modelling sessions were completed with 20 MRS stakeholders. Five meetings and five workshops were held at the beginning and end of each iteration, respectively.

IV. RESEARCH EXECUTION

This chapter describes the execution and results of the research.

A. Pre-study

The first step was to analyse the problem the company was experiencing with the MRS. The company representatives provided a data set containing descriptions of the reports, the reports' data structures and the stakeholders who were responsible for providing the reports. In addition, descriptions of the metrics in the MRS were provided, including metric input data, calculation formulas and the organisational units responsible for providing data. Currently, the metrics in question are reported both manually and with Excel templates.

After analysing the information, the scale and amount of the information presented from multiple stakeholder perspectives became the main problem. The first step was to select a way to structure all the information. A problem structuring method was determined to be a suitable framework since similar problems are generally approached in this way, a soft systems methodology (SSM) [28]. SSM was selected due its iterative approach and ability to conceptually model any stakeholder viewpoint into a *soft system model*. A soft system can be any system where both natural objects and humans interact, which is essential for SIS descriptions. SSM consists of:

- entering the problem situation,
- expressing the problem situation,
- formulating root definitions of relevant systems,
- building conceptual models of human activity systems,
- comparing the models with the real world,
- defining changes that are desirable and feasible, and
- taking action to improve the real-world situation.

Other problem structuring methods exist, such as multiview, information requirements analysis and logico-linguistic modelling. They are similar in approach to SSM but are designed to be more systematic and rigorous. In this case, because the principles remained the same, SSM was used to structure the MRS.

B. Action research iterations

Based on the pre-study, the action research began by structuring the problem situation so that it could be understood properly. SSM was selected as a starting point for the research activities. Table 1 describes the first two iterations.

 ISBN: 978-1-61208-438-1

TABLE I. ITERATIONS 1–2

Iter.	Diagnosis	Action planning and taking	Evaluation	Learning
1	Problem should be further modelled and analysed in order to understand it properly.	Create a soft system model from the MRS using SSM and the data obtained in the pre-study. Use a UML flowchart as a modelling language. Use Microsoft Visio as a modelling tool. Perform iteration lasting 3 months; one researcher creates the soft system model from MRS.	The soft system model resulting from SSM was found to be informative and easy to understand. The UML flowchart was able to model the information, and participants had experience using it. Microsoft Visio was able to model the soft system model.	The data given by the company in the beginning was insufficient, as the resulting soft system model in the end of the iteration was missing information. Stakeholders need to be involved directly. Data and functions were found to be missing after the soft system model was finished, especially manual work, which was done but is missing from the official documentation. A visual model of the soft system promotes communication and information distribution between stakeholders. The UML flowchart was sufficient for describing the MRS. It utilised familiar language and increased the acceptance of the method.
2	Stakeholders need to be directly involved to discover their understanding of the problem and the soft system where it resides. UML flowcharts and Microsoft Visio should still be used to visualise the soft system model. SSM should be used as long as the soft system model is able to present the problem. The modelling approach needs to be structured since data and activities can be hidden. An approach is needed to identify and model these data and activities.	Select two example metrics and relevant stakeholders in order to discover whether the soft system model is beneficial. Organise 1.5-hour modelling sessions with the identified stakeholders. One researcher and at least one stakeholder participate in each modelling session. The participating stakeholder must identify other stakeholders that can describe the soft system if the original stakeholder was not able to do so. Separate each stakeholder viewpoint with layers in Microsoft Visio. Utilise an input(s)–function(s)–output(s) structure for information flow to help identify what the stakeholders actually do in the system.	Using actual stakeholders helped create a soft system model that represented how the MRS worked in reality. Modelling sessions with each stakeholder allowed for the modelling and separation of stakeholder viewpoints within the same soft system model. Multiple viewpoints helped to remove uncertain parts from the model and increased the quality of data as they were refined and confirmed by more than one stakeholder. Stakeholders can be systematically added to the model by asking the stakeholders to identify who provides them with information or uses information provided by them. The layers used in Microsoft Visio clearly visualise how different stakeholders see, understand and work within the MRS. The input(s)–function(s)–output(s) structure of the model increased the level of detail and helped stakeholders work through the details of their work and the process.	The soft system model must allow for irregular and abstract viewpoint descriptions from different stakeholders, as it cannot be guaranteed that every stakeholder is able to use formal language. The input(s)–function(s)–output(s) structure should not be strictly enforced. Multiple viewpoints revealed variations and different data and activities than were originally known by the stakeholders. The differences and variations affected metric generation. A systematic method is needed, as multiple stakeholders are required to work on it simultaneously due to the scale of the problems and working environment. The soft system model was easy to understand, but the concept of SSM was not. It was seen as too vague, and it is unusable in its current form. Participants felt that the methodology, which helped them to approach problems in their work, was more valuable than solving the problem in the MRS. Participants discussed different problems related to MRS that became visible form the model, as they appeared to have even more value if solved. A need to analyse value from stakeholder's perspective surfaces as different changes or impacts were evaluated against the soft system model.

The first iteration concentrated on creating a conceptual model of the problem. Unified Modelling Language (UML) flowchart was selected as a basic modelling language. However, it quickly became obvious that a structure for the modelling approach was required to obtain an accurate model. Allowing abstract descriptions hid the information obtained by the stakeholders. Therefore, the second iteration utilised the input–function–output structure to describe any activity performed by the stakeholders. In addition, actual stakeholders were involved to obtain the data directly from the stakeholders themselves, as the original data was insufficient. The results of the second iteration indicated that the company's problem was not to just to solve the problems in MRS, but also how to analyse it systematically in SIS environment. It became clear the company personnel had to work in a certain manner and with certain restrictions caused by the SIS development. The participants felt that the methodology used in the research would be more useful than just solving the problem they had with the MRS.

It was determined that the actual goal of the research should be to design a methodology that the company could use to analyse problems during SIS development, as it was believed that the stakeholders would benefit from analysing the problems themselves. Since SSM was perceived to be too vague and unfamiliar to participants, it was designed to be part of a stakeholder analysis. Essentially, stakeholder analysis and SSM have similar outcomes. However, the participants understood the concept of stakeholder analysis better. Utilising a UML flowchart (UML was identified to be a suitable modelling language) and existing systematic approaches to stakeholder analysis (e.g., [22], [23]), three more iterations were run in order to develop and refine a stakeholder analysis (S3AM) suitable for an SIS development environment. Table 2 describes the following three iterations, which aimed to develop a methodology to analyse problems during SIS development.

Iterations 3–5 mainly concentrated on identifying the special attributes of SIS development and how the S3AM needs to support the company's development process. The result of the action research was an exact methodology, which is described in the next sub-chapter.

 ISBN: 978-1-61208-438-1

TABLE II. ITERATIONS 3–5

Iter.	Diagnosis	Action planning and taking	Evaluation	Learning
3	The fact that SSM creates soft system models helps individuals comprehend and approach problems during SIS development. The SSM approach adapted in the research should be methodised for the company. The SSM was perceived as too vague and difficult, and it was too general to be used as such in SIS environment. Stakeholder analysis is a better-known concept in software development and has similar outcomes to SSM. The solution was to combine stakeholder analysis and a framework for creating a soft system model from SSM. The value of introducing any change to the soft system should be analysed, preferably by asking the stakeholder directly or using a pre-defined value measurement.	Design a first version of the S3AM based on the findings of previous iterations utilising existing systematic stakeholder analysis approaches. Extend the analysis to a larger part of the MRS to test the S3AM in its intended environment. Select eight metrics reported by eight different product lines with responsible stakeholders to be analysed. Organise 1.5-hour modelling sessions with each stakeholder. One researcher and at least one stakeholder participate in each modelling session. Ask each stakeholder how many working hours they spend performing particular tasks to evaluate value.	Identifying additional stakeholders by asking participating stakeholders was effective. As the analysis was extended to a larger portion of the MRS, the analysis must be conducted by multiple persons. Getting stakeholders to participate is becoming difficult due to the number of stakeholders and scheduling problems. When stakeholders saw the existing model during modelling sessions, they seemed to understand the modelling approach, and in most cases, they described their activities related to MRS without requiring further instruction. This applied to both managers and engineers.	The variations and differences between data and activities become even more pronounced as more stakeholder viewpoints are added. Promoting the input(s)–function(s)–output(s)—language structure, which is typical in system descriptions, produced a richer and more detailed soft system model. Visualising the models increased the effectiveness of communication in every meeting. The existing soft system model helped participants understand their tasks and increased the speed of the modelling process faster as participants had an example to work from. Stakeholders felt that the ability to see other stakeholders' viewpoints in the model increased their understanding of how the MRS worked. This led to discussions on how the MRS could be improved and how it should be analysed to understand its problems.
4	Explicit visualisation of stakeholders' viewpoints was an eye-opener for many stakeholders. Stakeholders who can see and understand other viewpoints are able to evaluate impact and value from a wider perspective.	Continue the modelling performed in Iteration 3. Make the existing soft system model available and present it to stakeholders.	Adding more stakeholder viewpoints to the soft system model helped stakeholders evaluate impact and value from a wider perspective. Improving the soft system model makes interpretation and reading the model more difficult for stakeholders.	Model abstraction is needed in both the soft system model and in the tool used to generate the soft system model. Gradually building and constantly refining a soft system model supports the distribution of work and the co-operative nature of the development environment. Feedback from stakeholders signalled a need to identify the most important areas to analyse and where there is missing information in order to determine how the system should be changed to address problems.
5	Abstraction of stakeholders' viewpoints is necessary in large systems. The SIS environment causes certain restrictions and requirements for the S3AM because work is continuous and the aim of the analysis changes when new information is made available.	Model abstraction layers using descriptions of stakeholders' viewpoints. Analyse the original problem based on the soft system model and create a separate impact layer where the system automation can be evaluated. This information can be used to determine whether the current soft system model is adequate for determining the value of automation and how it should be implemented. Continue the modelling performed in Iterations 3 and 4.	The implementation layer allowed stakeholders to identify missing information and unclear areas. Stakeholders identified problems with more value than the original problem. The soft system model, along with the implementation layer, helped to direct the analysis based on unclear data that was connected to the impact model. Previously unknown but relevant stakeholders were identified based on the analysis in the impact layer due to missing information.	Multiple viewpoints increased the visibility and transparency of the MRS and refined the original problem. Multiple viewpoints allowed for the evaluation of different value perspectives, especially local and strategic perspectives. Constant analysis of the problem directed the modelling of the soft system and provided direction for the analysis based on the impact of the problem. When the impact extended out of the model, it was an indication for doing further analysis on those sections. The introduction of an analysis layer for analysing the impact of automation helped stakeholders identify missing or unclear information and find previously unknown stakeholders.

C. S3AM

The principle idea of S3AM is to model different stakeholder viewpoints into the same soft system model. Each viewpoint, however, is the conceptualisation of a stakeholder's understanding of the soft system. In essence, each viewpoint contains information about how the stakeholder perceives and understands the way in which a real life phenomenon works.

The S3AM starts with a single stakeholder, who identifies a soft system of interest. Due to this interest, the stakeholder intends to present a request, requirement, need or problem that aims to change that soft system in a certain manner. The stakeholder is seeking to provide information about how to change a certain soft system he believes requires such change.

In order to create the soft system model, a certain structure was required to describe stakeholder viewpoints. In this case, all soft system descriptions follow the same

principle of information flow description: input(s)-function(s)-output(s). Each stakeholder's viewpoint is constructed in a similar manner: information (documents, emails, communication, etc.) is received, something is done with the information (analysis, transfer, format changes etc.) and the result is a defined output (report, piece of information, emails, etc.). For example, a stakeholder is in charge of collecting summaries of metric data from two other stakeholders. Figure 1 demonstrates the soft system model created from the stakeholder's viewpoint.

From this stakeholder viewpoint, the summarised metric data element in the upper right corner and the summarised data element on the upper left form the boundaries of the stakeholder's viewpoint. The stakeholder does not know how the other stakeholders actually perform the data summaries and what data is used.

Now that the boundaries are known for the first stakeholder, the next step is to analyse the intent of the stakeholder (e.g., to automate data collection and create a data summary). The known soft system model is now analysed to determine whether it contains enough information to analyse the real impact from all relevant stakeholder viewpoints.

After the initial soft system model is created, the next step is either to analyse how the soft system would change if some desired change were introduced, or to simply extend the model by adding more stakeholder viewpoints. The first case asks practitioners to analyse how the current soft system changes if some desired change, for example a stakeholder's need, affects it. In this case, an impact layer is drawn using the existing soft system model modified by the change. If the change affects any of the boundary elements in the soft system model, the stakeholder whose viewpoint has the boundary element should identify a stakeholder who knows how the soft system worked prior to that element, and it should then be modelled. This continues until there are no boundary elements affected by the change. This helps to direct the analysis effort to those parts that are not yet modelled but will be affected by the desired change. The other option is to simply ask each stakeholder to identify other stakeholders that can describe the system beyond the boundary elements of his or her own viewpoint and keep extending the model.

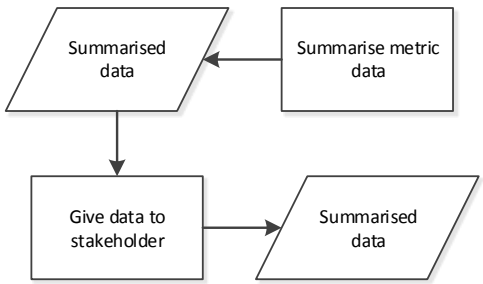


Figure 1. An example of a single stakeholder viewpoint

As more stakeholder viewpoints are added, there will eventually be shared elements and conflicting viewpoints. Figure 2. demonstrates an additional stakeholder viewpoint added to the soft system model. The colour white indicates a stakeholder who collects the metric data from the report, summarises it as a single figure and gives it to the original stakeholder. Another stakeholder asks for the metric data directly from the engineers, summarises it and gives it to the original stakeholder. However, the stakeholder also states that it is not exactly the same metric data; it is presented by a separate data element.

As the model is updated, it is important to verify with each stakeholder that existing elements (both functions and data) remain the same along with the data flows. If they are not, they must be modelled separately to highlight the differences. One of the key principles is that each stakeholder 'owns' his or her own system description. Thus, if any changes were made to a single viewpoint, the stakeholder who owns it had to accept the change. This was determined to be an important feature, as it prevents a loss of information by assuming the elements in the soft system are the same. Therefore, each element in the soft system model belongs to one or multiple stakeholders, and any change to an element requires all stakeholders to agree to the change or newly created element.

To analyse the value of any change to the soft system, a need presented by a stakeholder should be analysed and modelled as an impact model, describing a solution for the need. The impact model can be used to determine all elements affected by the impact and therefore track all the viewpoints in which those new, modified or deleted elements reside. The value can be analysed either by measuring impact to the work effort or by asking the stakeholder directly what value he believes the new soft system would bring.

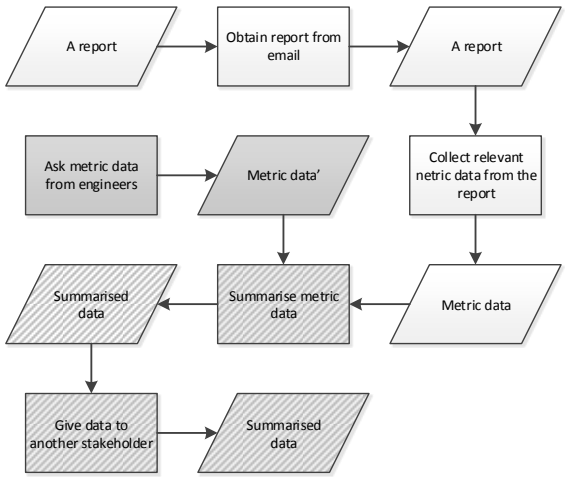


Figure 2. Two stakeholder viewpoints in a single soft system model

These values can be summarised to evaluate the overall impact. Furthermore, these values can also be used to redesign the solution to achieve a different impact with

 ISBN: 978-1-61208-438-1

different values. When the value is known for each stakeholder, it is possible to determine the overall effect and analyse the effect for different groups of stakeholders. Values can be both negative and positive.

Based on the results, it was determined that using a modelling language with which the organisation is familiar supports the usability of the method. Therefore, S3AM allows utilisation of any modelling language as long as it supports visualisation, an input(s)-function(s)-output(s) model structure and allows attributes to be defined for each element. The important factor, however, is the ability to visualise the soft system and only promote the input(s)-function(s)-output(s) modelling structure to help stakeholders structure their viewpoints. This allows practitioners with varying levels of technical understanding to understand the described soft system more quickly and from different perspectives.

V. DISCUSSION

The results from the action research show that problems in which humans are an integral part are hard to describe, comprehend and communicate. Understanding and structuring such a problem requires human understanding. The results further demonstrate how the same phenomena, the MRS and data used by the system, are perceived and understood in a different manner by each stakeholder. S3AM was developed as a combination of SSM principles, stakeholder analysis and a model-based approach to a soft system where humans play an integral role. These systems are never plain, hard systems. They are, rather, a collection of hard systems interacting with themselves or with humans. While they can be described using a hard system approach, the difference is that hard systems are considered to be free of interpretation and follow almost exact rules. Such systems can be described with exact languages due the nature of them. However, when humans are added to these systems as actors or observers, the way the system is perceived and how it functions now depends on individual perception, understanding and behaviour. Multiple actors can understand and describe same phenomena in equally different ways, all of which need to be captured somehow. For example, a single product in the telecommunication domain can simultaneously include building the physical network, maintenance and customer support. Due to the nature of human thinking, the same principles do not apply to designing, analysing and modelling, and a 'soft' approach is required. While the goal of a soft system approach is the same as a hard system approach (i.e., model a real world phenomena), the soft system approach introduces a way to capture and understand different viewpoints of human actors.

S3AM adapted the systems thinking part from SSM, where root definitions of the soft system in question are determined and modelled using semi-formal language. The key principle is to allow highly abstract, conflicting, very structured schematics or any other forms of system descriptions to exist in a single soft system model. In S3AM, this was done by utilising stakeholder viewpoints as layers and the boundaries of these viewpoints as connection points to other viewpoints. While this model itself was not coherent, the main idea was to analyse and connect models to form a coherent and structured soft system model. This allows different worldviews to come together and facilitate consensus building between stakeholders. At the same time, the method gradually removes differences in the concepts and terms between stakeholders, who are able to see them through the viewpoint of others and obtain crucial insight into how others think. Workshop discussions indicated problems in communication between management and engineering. The ability to have both viewpoints in the same model helped stakeholders from both groups communicate more efficiently and allowed them to understand each other's concerns and perspectives.

S3AM fulfilled the role of stakeholder analysis by allowing participants to identify relevant stakeholders, elicit and analyse their needs and have the means to verify and validate the problem system. In the end, S3AM was designed to be simplistic and direct the user to structure any problem as a soft system model, utilising stakeholder viewpoints to describe it as accurately as possible. It also provides information in the form of impact and value to facilitate understanding of the requirements, negotiation and agreement on a solution. The S3A also addresses the identified issues within stakeholder analysis by providing a systematic and defined approach and analysis method. It allows systematic construction of a soft system model explaining how different stakeholders perceive the real world. A local problem was structured and expanded to describe the soft system from additional perspectives, especially a strategic perspective, which allowed identification of problems and issues that potentially bring more value to multiple stakeholders rather than a few local stakeholders.

However, the research also showed that systematic and defined stakeholder analysis alone is not enough. How it is implemented and how it creates new information for the process is equally important. This defines the usefulness of the method and justifies its existence. Stakeholder analysis in SIS development is not only about identifying the requirements and agreeing on their specification through negotiation, prioritisation and validation. It also communicates information, creates awareness and elevates thinking to higher abstraction layers, enabling the discovery of problems or issues that create strategic value. S3AM enabled discovery and analysis of impacts and values outside the original problem description. In such a situation, the original requirement only presented a situational or tactical problem. Analysing it systematically from different stakeholder viewpoints revealed 'strategic' problems that were previously unknown. As local and strategic perspectives were visible, S3AM had a clear impact on removing uncertainty within the participants. It effectively

 ISBN: 978-1-61208-438-1

increased the quality of the information that described the soft system. Essentially, this helped the stakeholders evaluate how much information they had and what information was potentially missing. Furthermore, as more stakeholders shared a viewpoint, the quality of the information increased.

The action research also revealed the needs and limitations of the SIS development process. This had a clear impact on the usability of the method. The case company's development process leans towards a decision-oriented [29] culture, emphasising the nature of information needed in the process. Furthermore, the need to make decisions in quick intervals was also apparent, and some kind of result was always necessary to either satisfy the information need to make a decision or to continue the analysis, as the risk of the unknown was too great. In this sense, stakeholder analysis also needs to inform the practitioners whether they know enough or whether there is missing information that still must be analysed. The process, methods and practices, as well as the workflow used to implement and enact them, should support freedom in the order of activities and the implementation of practices and strategies. In the modern world, stakeholders come from different organisations, forming distributed teams that work with the help of communications technology. In distributed teams, people work as dynamically and spontaneously assembled groups in a collaborative mode [7]. However, the developer's activity is still guided by objectives, work requirements, constraints and resources, which form the fundamental constraints on workers' behaviour [30]. Software design is never a fully rational process [31][32] as:

- People who commission software system do not know exactly what they want and frequently are unable to elaborate what they want.
- Even if the real need is known, further information needs surface as development moves towards implementation.
- Most humans cannot comprehend all information, even if all information is available.
- Only the most trivial projects are not subject to external changes.
- Human errors can only be avoided by excluding humans in the development.
- Preconceived ideas often influence the design process in ways that are not necessarily appropriate.
- There may be a reuse of software developed by others or from other projects that is not necessary ideal.

For the SIS development, gradual expansion, refinement and correction of the entire soft system model was a practical approach. The complexity and uncertainty in the beginning required that the problem first be structured and the data refined to validate it. The ability to modify any part at any given time was seen as an important aspect of the method. Since the information was not complete in the beginning (or it could not be properly comprehended), validation from multiple viewpoints was also essential.

The participants saw visualisation as an essential feature. They frequently talked about the same system but tried to explain the differences they perceived. They lacked either the words or expressions to describe this effectively for other stakeholders. However, when each viewpoint was modelled and the entire soft system was visible in a single model, the differences were communicated to each stakeholder more easily.

VI. THREATS TO VALIDITY AND LIMITATIONS

The reliability of the data and results was ensured via a rigorous research protocol with peer reviews by researchers and company representatives. The action research cycles were described and followed throughout the research. The modelling sessions were recorded and transcribed by the researchers.

This study is limited only to the telecommunication domain. Furthermore, only one company was involved in this research, limiting the generalisability of the results. However, the study uses a well-established problem structuring method that has been used in multiple domains. In order to make definitive conclusions, more domains and companies should be involved in future research.

The way the action research was implemented in this study also introduces a danger of positive bias within researchers and company participants. Due to the constant communication and interventions in the company, participants could be positively biased, producing only positive results. This issue was addressed by having multiple different viewpoints presented in the meetings. In addition, agreement over clear roles and rigorous research methods helped participants remain observers.

VII. CONCLUSIONS

In this paper, action research was conducted in a telecommunication domain company, aiming to create a stakeholder analysis for SIS development. Using the SSM as a starting template, the result was a Soft System Stakeholder Analysis Methodology (S3AM).

The main contribution of this paper is that it shows the importance of systematic analysis of stakeholder viewpoints from a soft system perspective. Furthermore, it raises the importance of method usability, a factor that cannot be ignored in SIS development as it directly affects data collection, quality and analysis. From an academic perspective, this study provides industry insight in terms of stakeholder analysis and utilisation in an SIS environment. It demonstrated the importance of systematic problem structuring and model creation in stakeholder analysis. Finally, the results present a systematic and practical approach for stakeholder analysis in an SIS development environment.

 ISBN: 978-1-61208-438-1

Future research should examine the use of S3M in domains other than telecommunication to verify and validate the methodology and its generalisability. In addition, more research is needed concerning the development of the modelling language.

ACKNOWLEDGMENT

The authors would like to thank the Digile N4S project and TEKES for providing support and funding for the research.

REFERENCES

[1] M. Broy, "The 'grand challenge' in informatics: Engineering software-intensive systems," Computer, vol. 39, no.10, 2006, pp. 72–80.

[2] Recommended Practice for Architectural Description of Software-intensive Systems, IEEE-Std-1471-2000. New York, USA, October 2000.

[3] H. Giese and S. Henkler, "A survey of approaches for the visual model-driven development of next generation software-intensive systems," Journal of Visual Languages and Computing, vol. 17, December 2006, pp. 528–550.

[4] P. Robillard, "The role of knowledge in software development," Communications of the ACM, vol. 42, num. 1, 1999, pp. 87–92.

[5] F. O. Bjørnson and T. Dingsøyr, "Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used," Information and Software Technology, vol. 50, 2008, pp. 1055–1068.

[6] S. Buckingham Shum, "Negotiating the Construction of Organizational Memories," in U. Borghoff and R. Parechi, Eds. Information Technology for Knowledge Management, Berlin, Germany: Springer Verlag, 1998, pp. 55–78.

[7] W. Prinz, H. Loh, M. Pallot, H. Schaffers, A. Skarmeta, and S. Decker, "ECOSPACE – Towards an Integrated Collaboration Space for Eprofessionals," in International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2006, pp. 39–45.

[8] J. Rosenhead and J. Mingers, J, Rational Analysis for a Problematic World Revisited. Chichester: Wiley, 2001.

[9] G. Kotonya and I. Sommerville, Requirements engineering: Processes and Techniques. Chichester: Wiley, 1998.

[10] I. Sommerville, Software Engineering, 7th ed. Harlow: Pearson Education Limited, 2004.

[11] P. Robillard, P, "The role of knowledge in software development," Communications of the ACM, vol. 42, no. 1, 1999, pp. 87–92.

[12] F. O. Bjørnson and T. Dingsøyr, "Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used," Information and Software Technology, vol. 50, 2008, pp. 1055–1068.

[13] L. J. Bannon and K. Schmidt, "CSCW: Four Characters in Search of a Context," Proceedings of the First European Conference on Computer Supported Cooperative Work (ECSCW 89), Gatwick, London, 13–15 September 1989, pp. 358–372.

[14] J. McKernan, Curriculum Action Research: A Handbook of Methods for the Reflective Practitioner, 2nd ed. London: Kogan Page, 1996.

[15] R. E. Freeman, Strategic Management: A Stakeholder Approach. Boston: Pitman, 1984.

[16] R. E. Freeman, J. S. Harrison, and A. C. Wicks, Managing for Stakeholders: Survival, Reputation, and Success. Yale University Press, 2008.

[17] A. P. Friedman and S. Miles, Stakeholders, Theory and Practice. Oxford: Oxford University Press, 2006.

[18] L. Ballejos and J. Montagna, 2008, "Method for stakeholder identification in interorganizational environments," Requirements Engineering, vol. 13, no. 4, pp. 281–297.

[19] A. D. Davis, Just Enough Requirements Management: Where Software Development Meets Marketing. New York: Dorset House Publishing, 2005.

[20] I. Alexander and R. Stevens, Writing Better Requirements. Reading: Addison Wesley, 2002.

[21] B. Boehm, B. Bose, E. Horowitz, and M. J. Lee, "Software Requirements Negotiation and Renegotiation Aids: A Theory-based Spiral Approach," Proceedings of the 17th International Conference on Software Engineering, 1995, pp. 243–253.

[22] J. McManus, 2004, "A stakeholder perspective within software engineering projects," Engineering Management Conference, vol. 2, pp. 880–884. IEEE.

[23] S. L. Lim, D. Quercia, and A. Finkelstein, "StakeNet: Using Social Networks to Analyse the Stakeholders of Large-scale Software Projects," Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 10), 2010, pp. 295–304.

[24] R. McTaggart, "Guiding principles for participatory action research," in Participatory Action Research: International Contexts and Consequences, R. McTaggart, Ed. Albany: State University of New York Press, 1997, pp. 25–43.

[25] R. Baskerville and A. T. Wood-Harper, "A critical perspective on action research as a method for information systems research," Journal of Information Technology, vol. 11, no. 3, 1996, pp. 235–246.

[26] R. Rapoport, "Three dilemmas in action research," Human Relations, vol. 23, no. 6, 1970, pp. 499–513.

[27] G. Sussman and R. Evered, "An assessment of the scientific merits of action research," Administrative Science Quarterly, vol. 23, 1978, pp. 582–603.

[28] P. B. Checkland, Systems Thinking, Systems Practice, John Wiley & Sons Ltd., 1981.

[29] J. Hyysalo, M. Kelanti, J. Lehto, P. Kuvaja, and M. Oivo, Software Development as a Decision-Oriented Process. Switzerland: Springer International Publishing, 2004.

[30] J. Rasmussen, A. M. Pejtersen, and K. Schmidt, Taxonomy for Cognitive Work Analysis. Roskilde, Denmark: Risø National Laboratory, Risø report M-2871, 1990.

[31] D. L. Parnas and P. C. Clements, "A rational design process: How and why to fake It," IEEE Transactions on Software Engineering, vol. SE-12, no. 2, February 1986, pp. 251–257.

[32] D. L. Parnas, "Document based rational software development, knowledge-based systems, vol. 22, Elsevier, 2009, pp. 132–141.

[33] I. Alexander and S. Robertson, "Understanding project sociology by modeling stakeholders," IEEE Software, vol. 21, no. 1, 2004, pp. 23–27.

 ISBN: 978-1-61208-438-1

Publish/Subscribe Cloud Middleware for Real-Time Disease Surveillance

Silvino Neto, Márcia Valéria, Plínio Manoel, Felipe Ferraz
Recife Center for Advanced Studies and Systems (CESAR)
Recife – PE, Brazil
e-mail: {silvino.neto, marciavr.souza, ti.plinio}@gmail.com, fsf@cesar.org.br

Abstract - **This paper presents the design and implementation of a cloud-based middleware built on top of the Google Cloud Platform (PaaS), in order to exchange real-time information about outbreak notifications of global diseases in a system-level by using an extension of the HL7 Fast Healthcare Interoperability Resources (FHIR) specification to support statistical data based on the ICD-10 medical classification list. The proposed solution aims to allow healthcare organizations to register their systems to send and receive notifications, so the alerts are spread to all the subscribed systems using webhooks in a publish/subscribe fashion.**

Keywords - middleware; google cloud pub/sub; google cloud platform; FHIR

I. INTRODUCTION

There is an increasing demand for real-time monitoring of a broad variety of complex events. The processing of these information streams originated from multiple sources allows the early identification of threats and swift response. With the advent of modern communication technology, we are able to report incidences of disease outbreaks worldwide in a timely manner. Institutions such as the World Health Organization (WHO) and the Centers for Disease Control have been involved in the development of surveillance mechanisms that triggers alerts that support the decision-making process about how to respond to these incidents.

As results, there has been many successful experiences in using different forms of communication to exchange data related to surveillance and control of diseases, such as Short Message Service (SMS) [1][2], integration of device data capture [3] and system-level notifications [4].

Healthcare records are increasingly becoming digitized. In order to support system-level exchange of clinical data, a set of standards are required. The HL7 specification comprises a set of international standards to exchange clinical data between healthcare applications. In an attempt to improve its simplicity and extensibility, the HL7 introduced a new specification known as Fast Healthcare Interoperability Resources (FHIR). When compared with its predecessors, HL7 FHIR offers a whole new set of features, such as: support for multiple data formats: Extensible Markup Language (XML) and JavaScript Object Notation (JSON), extensible data model and a RESTful API.

This paper describes the Platform for Real-Time Verification of Epidemic Notification (PREVENT), a cloud-based message-oriented middleware in collaboration with the use of an extended instance of the FHIR specification to support statistical reports for disease surveillance in order to monitor and notify outbreak occurrences in real-time fashion. In our solution, we have developed our middleware application on top of the Google Cloud Platform, using the Google Cloud Pub/Sub, which is a many-to-many, asynchronous messaging service. Healthcare organizations may send and receive push notifications through the use of a registered webhook endpoint that can accept POST requests over HTTPS.

This paper is further structured as follows: In Section 2, we discuss the foundations for this paper. In Section 3, we present the architectural approaches proposed for the middleware and some of the design choices implemented. In Section 4, we explain our evaluation approach and present the results obtained. In Section 5, we discuss related work and finally, Section 6 presents our conclusions and possible future work.

II. FOUNDATIONS

In this Section, this paper presents key concepts that served as basis for the development of this work.

A. WHO

The WHO is a specialized worldwide health agency subordinated to the United Nations (UN) that, according to its constitution [5], one of its main objectives is the development and improvement of the health of people to the highest possible levels. Still, according to the WHO constitution, it is responsible for coordinating efforts to control and prevent outbreaks and diseases. The WHO supervises the implementation of the International Health Regulations and publishes a series of medical classifications, including the International Statistical Classification of Diseases and Related Health Problems (ICD) [6]. The ICD is designed to promote international comparability in the collection, processing, classification, and presentation of mortality and morbidity statistics.

According to the International Health Regulations (IHR), an international legal instrument that is compulsory in 196 countries and in all the WHO member states, its goal is to assist the international community in the prevention and response to potential cross-border public health risks. The IHR requires that countries report disease outbreaks and public health events to WHO [7].

The present work discusses a system platform that allows national health organizations, members of the United Nations, hospitals or healthcare agencies, regardless the location, to subscribe their applications to send and receive real-time notifications for disease surveillance. Thus, they contribute with the propagation of the notified information, so it can achieve the widest possible reach through the use of

 ISBN: 978-1-61208-438-1

a cloud-based platform. Therefore, countries and healthcare organizations may act promptly under the emergency and disaster risk management protocol to prevent, prepare, respond and recover from incidents due to any danger that might represent a threat to human health security.

B. HL7 and FHIR

To reach its goal, this work analyzes a set of international standards that provide a framework for the integration and share of clinical and administrative data between systems and electronic devices dedicated to health care. The HL7 [8] was created in 1987 and has been maintained by Health Level Seven International, a nonprofit international organization that supports and promotes the development of international interoperability standards in healthcare systems.

The second version of HL7, an *ad hoc* approach to integrate various fields in health care, hospitals, clinics and administrative applications, has become a widely used standard, adopted and supported by most healthcare application vendors in North America [9]. Despite HL7 v2 wide acceptance, the limitations of the *ad hoc* approach have not allowed significant high scale use in larger multiplatform environments. Another downside observed on HL7 v2 is the lack of a formal data model that can unify concepts and interfaces for message transmission. HL7 v3 emerged as a response to all the problems recognized on the previous version. However, it was heavily criticized by the industry for being inconsistent, overly complex and infeasible to implement in real life systems. For a while, it appeared as if interoperability initiatives for health care had lost momentum.

Hence, FHIR was created with the objective of improving HL7 messaging standards and addressing some of the issues identified on the previous specifications. There have been discussions towards a new approach for data exchange in health care. This approach provides a Representational State Transfer (REST) interface, which is a very simple and lightweight interoperable alternative for system integration. REST-based architectures are known for its scalability, user-perceived performance and ease of implementation approach that provides a fast data transmission pattern mostly using the HTTP protocol [10]. Resource interoperability allows information to be readily distributed and provides an alternative to document-centric approaches by directly exposing data elements as services.

FHIR uses syntax based on XML or JSON, simplifying the system-level communication. It also offers support for an extensible data model, allowing applications to enhance its data structures using FHIR extensibility mechanism. The features mentioned on this paper were decisive factors for the adoption of FHIR on the development of PREVENT.

C. Cloud Computing and Scalability

According to A. T. Velte et al. “In essence, cloud computing is a construct that allows you to access applications that actually reside on a location other than your computer or other Internet-connected device; most often, this will be a distant data center” [11]. This is a constant reality for the majority of the Internet users on a daily basis. Among the benefits of cloud computing cited by [11], there are simplicity, knowledgeable vendors, more internal resources, security, and scalability.

Scalability is seen as a fundamental feature of cloud computing. It appears as if computational resources are infinite and end users easily notice the increase of performance of used resources in a cloud-based platform. Scalability is not restricted to expanding resource capacity, being scalable is to increase the capacity of operations in an efficient and adequate manner, maintaining the quality of service [12]. In the literature, it is possible to identify two dimensions of scalability: vertical and horizontal. Vertical scalability refers to the improvement of hardware capabilities by incrementing existing nodes individually. Horizontal scalability refers to the addition of extra hardware nodes to the current solution in a way that it can be possible to distribute application requests between these machines [13]. In the context of this work, PREVENT is designed and implemented with focus on horizontal scalability. In order to sustain a large volume of time-constrained notifications and to leverage the platform overall scalability, PREVENT is deployed in a cloud-based platform.

D. PREVENT and Complex Event Processing (CEP)

CEP is a new technology to extract information from distributed message-based systems. This technology allows users of a system to specify the information that is of interest to them. It can be low-level network processing data or high-level enterprise management intelligence, depending on the role and point of view of individual users. It operates not only on sets of events but also on relationships between events [14].

In order to respond in a suitable manner, it is fundamental to use technology that supports the construction and management of event-oriented information systems, and is also able to perform real-time data analysis. CEP consists in processing various events in order to identify their significance within a cloud of information [15]. CEP involves rules to aggregate, filter and match low-level events, coupled with actions to generate new, higher-level events from those events [16].

PREVENT has its own complex event processing unit, namely, PREVENT CEP Engine (PCEPE). PREVENT randomly receives data messages derived from healthcare applications subscribed as data providers or data sources, once data messages have been received, they are delegated to PREVENT internal complex data processing unit (PCEPE) that identifies the source and semantics of the data received, extracting relevant information.

After the information extraction phase previously described, the data collected goes through a second-phase analysis that intends to identify if the events notified at that time indicate a warning situation. As an example, a significant volume of reports of a certain disease from a specific geographically delimited area, points to a relevant situation of alert.

Finally, PREVENT only delivers relevant notifications to each subscribed message receiver. Figure 1 presents the

 ISBN: 978-1-61208-438-1

processing flow for data received from health care organizations in the PREVENT platform.

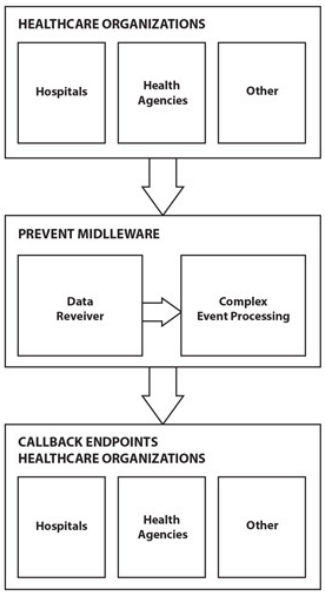


Figure 1. CEP diagram in PREVENT.

An event processing approach is ideal for applications concerned with the constant delivery of responses [15]. For sensitive information that requires a high level of consistency, it is extremely important that PREVENT responses are cohesive. This way, having an internal unit for the processing of the received events is required for the proper functioning of the PREVENT platform.

III. MIDDLEWARE

In this Section, this paper explores the proposed architecture and the workflow of events implemented in our middleware platform. Every step below is described as part of the process designed to perform the management of the subscribed applications, the process of data analysis and the delivery of notifications in the subscription topic:

- Healthcare applications may subscribe to our middleware platform in order to send and receive notifications to/from other systems;
- PREVENT will register the healthcare application in a subscription topic, and reply with an assigned application ID;
- Healthcare applications may now send notifications to PREVENT subscription topic;
- PREVENT will perform a real-time analysis of the data received, and publish notifications that match the specified criteria;
- Healthcare applications may now receive asynchronous notifications sent by other applications, delivered by PREVENT in a push request.

A. System Architecture

The system architecture designed for PREVENT is illustrated in Figure 2. In this diagram, PREVENT is organized into the Google App Engine [17] which is a hosting environment for web-based cloud applications. It is part of the Google Cloud Platform, as well as the Google Cloud Datastore, which is a schemaless NoSQL scalable datastore, and the Google Cloud Pub/Sub, an asynchronous messaging framework. Both framework platforms are used for data persistence and messaging (publish/subscribe pattern) services. It is important to mention that this middleware was developed in the Java programming language, using the Java Servlet API, a standard to implement applications hosted on Web servers under the Java platform. Despite being implemented on top of the Java platform, PREVENT is a completely agnostic technology, it uses interoperable standards such as HL7 FHIR, REST and JSON, and it can be integrated to any healthcare application, regardless the implementation language.

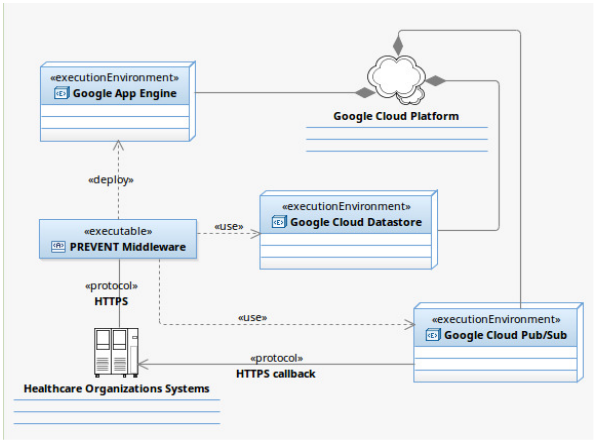


Figure 2. PREVENT Deployment Diagram.

Persistence is used on PREVENT to store data related to the subscribed healthcare applications and the statistical reports extracted from notifications received. The information stored by PREVENT at the Google Cloud Datastore is relevant not only for the delivery of real-time notifications, but it may also be useful for audit and access control policy and procedures. The data stored is replicated across multiple datacenters using a highly available platform based on the Paxos algorithm, which is a family of protocols for solving consensus in distributed environments [21].

PREVENT is designed to be a Message-oriented middleware (MOM) platform. It is implemented by using the publish/subscribe pattern, since it requires many-to-many communication. The Google Cloud Pub/Sub [18] platform supports two delivery strategies: push and pull delivery. In the push delivery, the server sends a request to the subscriber application at a previously informed endpoint URL for every message notification. In the pull delivery, the subscribed

 ISBN: 978-1-61208-438-1

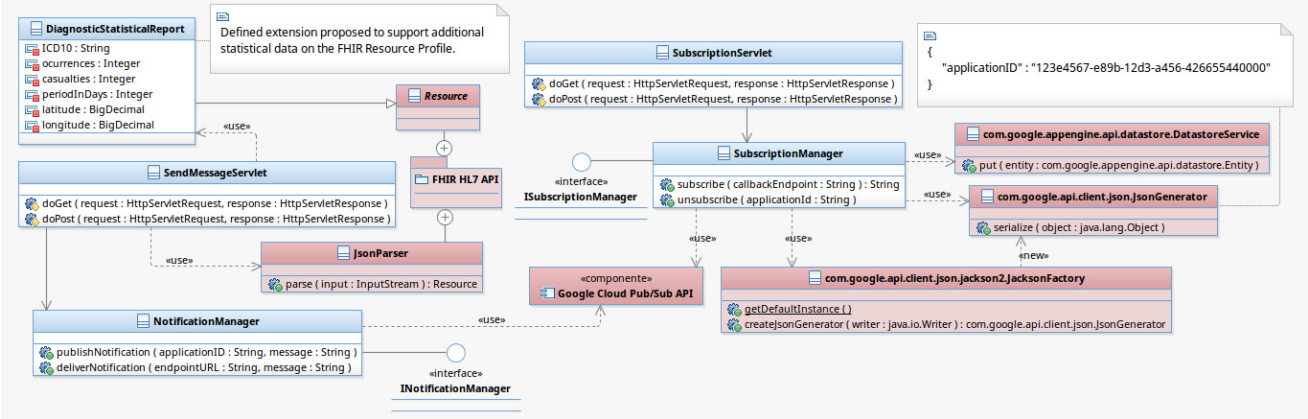


Figure 3. Subscription & Notification Scenario Class Diagram.

application has to explicitly invoke the API pull method, which requests the delivery of any existing message available in the subscription topic to the invoker. In our middleware implementation, we have chosen the push delivery strategy, based on the following criteria:

- Reduced network traffic;
- Reduced latency;
- Restrict/eliminate impacts of adaptation on healthcare applications (No need for message handling and flow control).

However, it's important to note that Google Cloud Pub/Sub platform is still a Beta release, so few limitations may be applied. Currently, the only supported endpoint URL for push delivery is an HTTPS server that can accept Webhook delivery. For this reason, we designed an internal HTTPS endpoint message listener that can be used as a proxy by healthcare applications in order to receive notifications that are subsequently forwarded to a regular HTTP endpoint URL.

B. *Subscription Request*

Healthcare organizations that want their application to send and receive notifications from our middleware should send subscription requests to PREVENT informing a single parameter named callback endpoint. The value of this parameter should correspond to a regular HTTP or HTTPS URL that will be invoked for notification delivery. As a response, PREVENT will reply with a unique application ID assigned for the request in JSON format as illustrated in Figure 3. As exhibited in the class diagram (See Figure 3), our middleware platform receives subscription requests through a Java servlet. The requests are subsequently assigned to the SubscriptionManager class, responsible for interacting with the Google Cloud Pub/Sub and Google Datastore APIs in order to both create a new subscription and store application data.

Once successfully registered, healthcare applications are allowed to send notifications to our middleware by informing its unique application ID with an extended HL7 FHIR message instance in JSON format.

C. *Publishing Notifications*

Previously registered systems should be able to publish notifications to all the subscribed applications. In order to do so, subscribed applications should always inform their application ID with an extended version of the HL7 FHIR message, as shown on Figure 3. As demonstrated in the class diagram (See Figure 3), there is another servlet class to receive requests for notification dispatch. This servlet class is expecting an HL7 FHIR message in JSON format. After the message is successfully parsed into its Java object representation, it will be dispatched to the *NotificationManager* class, responsible for the delivery of the message by invoking a method on the Google Cloud Pub/Sub API to add the new message to the subscription topic, making it available for delivery.

As mentioned earlier, HL7 FHIR provides a flexible mechanism for the inclusion of additional information into the FHIR data model. The class *DiagnosticStatiscalReport* shown in the diagram above (See Figure 3) is an example of an extension implemented on top of the FHIR specification. According to the FHIR specification, in order to use an extension, we must follow a three-step process, as defined in [19].

D. *Delivery of Notifications*

Notifications added to the subscription topic will be asynchronously sent to the registered endpoints for every subscribed application, in a multithread context. Messages are dispatched in the body of an HTTP push request. The body is a JSON data structure as depicted on Figure 4.

```
{
 "message": {
  "attributes": {
   "string-value": "string-value",
   // ... more attributes
  },
  "data": "base64-no-line-feeds-variant-representation-of-payload",
  "message_id": "string-value"
 },
 "subscription": "string-value"
}
```

Figure 4. Google Cloud Pub/Sub JSON message data structure.

 ISBN: 978-1-61208-438-1

The data attribute contained into the message data type structure holds the HL7 FHIR message encoded in Base64 format. Once the FHIR message is retrieved, in order to restore it back to its original form, healthcare applications must decode it. Furthermore, to indicate the successful delivery of the message received and avoid duplicate deliveries, healthcare applications must return one of the following HTTP status code: 200, 201, 203, 204 or 102. Otherwise, the Google Cloud Pub/Sub retries sending the message indefinitely to assure its delivery, using an exponential backoff algorithm in order to avoid network congestion [23]. The use of an exponential backoff algorithm is a feature offered by the messaging platform, in order to guarantee message delivery in case of message destination is unreachable or unavailable. Therefore, no message expiration or timeout is applicable.

It is expected to configure push endpoints with SSL certificates, so data integrity is guaranteed since all messages sent to them are encrypted over HTTPS. However, healthcare applications that do not provide an HTTPS webhook enabled endpoint, may still receive notifications using a regular HTTP endpoint URL. As already mentioned, PREVENT offers an internal message listener component that acts as a proxy for notification of delivery. During the subscription request processing, PREVENT will automatically assign an internal push endpoint for healthcare applications that informed regular HTTP URLs. Therefore, for every notification to be delivered, PREVENT hands it over to its internal message listener component that subsequently dispatches the notifications to their corresponding destinations. In order to manage HTTP connections efficiently, PREVENT uses the Apache HttpComponents, which is a toolset of low-level Java components APIs focused on HTTP [20]. Apache HttpComponents is designed to be a flexible framework, supporting blocking, non-blocking and event driven I/O models. In our middleware implementation, we selected the asynchronous non-blocking I/O model, in order to be able to handle thousands of simultaneous HTTP connections in an efficient manner.

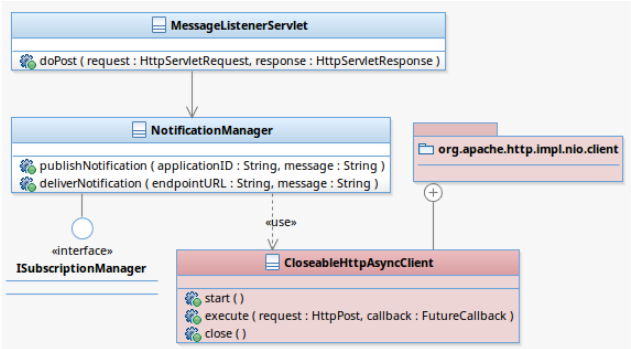


Figure 5. Message Listener Component Class Diagram

The class diagram shown on Figure 5 presents a short representation of the elements discussed on the previous paragraph.

IV. EVALUATION

We have developed a proof-of-concept implementation in order to evaluate the middleware. Our goal is to show how effective and responsive a cloud-based middleware platform for real-time surveillance can be. In accordance with the criteria established, we use a few metrics to measure the efficiency and performance of the middleware platform. In our experiments, we evaluate the middleware by using a set of simulation tools. The test environment set up for the evaluation is composed by a cloud-based instance of the middleware distributed into the Google Cloud Platform, a set of 50 callback endpoints, and an Intel i5 2.60GHz 6GB RAM Linux workstation. Each callback endpoint simulates a subscribed healthcare application, previously registered on the middleware. In order to act as an enlisted application, we have implemented a simple Java servlet class and a PHP file that basically returns an HTTP status code of 200 (OK) to acknowledge the successful reception of notifications delivered. The callback endpoints are deployed into two separate cloud platforms: Digital Ocean [22] and Google Cloud Platform. The tests are divided in two different scenarios. The first test scenario is executed at the execution environment of callback endpoints. On the second test scenario, we use a local computer workstation to perform a stress test. Both test scenarios will be conducted using a set of preconfigured simulation tools. In this evaluation, the following metrics are gathered and further analyzed:

- Throughput: measured by the number of delivered notifications per second;
- Error Ratio: measured by the number of requests failed or rejected by the middleware;
- Message Loss Ratio: measured by the proportion between the number of lost messages and the total number of messages delivered;
- Message Delivery Time: measured by the time taken to a notification request to be sent to the middleware and received by the subscribed healthcare applications.

A. *Simulation Description*

We perform the dispatch of messages using JMeter, a Java based testing tool, to send multiple HTTP requests.

On the first test scenario, our test suite is configured to send one message request per second, limited to 60 messages to be delivered to 50 subscribed callback endpoints. Messages used in the test are defined on HL7 FHIR JSON format. Each message size is approximately 1 KB. This scenario seeks to evaluate the performance of the middleware about the delivery ratio and message delivery time. In order to measure the total message delivery time, we must consider that the actual delivery of each message sent to the middleware occurs in an asynchronous manner. Therefore, time tracking has to be split into two separate variables:

- T^1 = Amount of time it takes for message requests sent to the middleware to be responded;

 ISBN: 978-1-61208-438-1

```
66.249.65.32 - - [05/Jun/2015:14:31:04 -0400] "POST /ReceiveNotification HTTP/1.1" 200 31 "-" "Mozilla/5.0 (compatible; Google Frontend/1.0)"
66.249.65.32 - - [05/Jun/2015:14:31:04 -0400] "POST /ReceiveNotification HTTP/1.1" 200 31 "-" "Mozilla/5.0 (compatible; Google Frontend/1.0)"
66.249.65.32 - - [05/Jun/2015:14:31:05 -0400] "POST /ReceiveNotification HTTP/1.1" 200 31 "-" "Mozilla/5.0 (compatible; Google Frontend/1.0)"
66.249.65.32 - - [05/Jun/2015:14:31:05 -0400] "POST /ReceiveNotification HTTP/1.1" 200 31 "-" "Mozilla/5.0 (compatible; Google Frontend/1.0)"
66.249.65.32 - - [05/Jun/2015:14:31:05 -0400] "POST /ReceiveNotification HTTP/1.1" 200 31 "-" "Mozilla/5.0 (compatible; Google Frontend/1.0)"
66.249.65.32 - - [05/Jun/2015:14:31:06 -0400] "POST /ReceiveNotification HTTP/1.1" 200 31 "-" "Mozilla/5.0 (compatible; Google Frontend/1.0)"
```

Figure 6. HTTP messages delivered

- T^2 = The average of the amount of time it takes for the middleware to send notifications received to all of the subscribed callback endpoints and get their message acknowledged;
- This way, Total Message Delivery Time = $T^1 + T^2$.

In order to obtain the values required to calculate both variables T^1 and T^2, we use both JMeter Test Results Report and Apache log4j, which is a Java-based logging utility.

The second test scenario is divided into 4 different test cases. Each test case has different settings, related to the number of messages that are expected to be processed. The number of concurrent threads (publishers) configured for each test case are limited to 10, 20, 60 and 100, respectively. Every thread is expected to simultaneously send one message per second. Every message received needs to be processed and delivered to 50 subscribed endpoints. Therefore, we expect a total of 14500 messages delivered after all test cases are completed. To evaluate message delivery to all of the subscribed callback endpoints, we created a shell script using AWK, which is a data-driven scripting language used in Unix-like operating systems, to extract and parse NGINX and Apache HTTP Server access logs in order to quantify the number of successfully received requests based on a pre-determined pattern as shown on Figure 6.

The results obtained will be summarized and compared to the total amount of messages sent. As a result, we expect to evaluate the middleware performance under heavy load, collecting and analyzing metrics like the throughput and message loss ratio.

B. *Results*

The evaluation shows that a cloud-based middleware for real-time surveillance works reliably and efficiently to report critical events in a timely manner. As illustrated on Figure 7, from a total of 14500 messages sent to the middleware platform, we reported 356 failed or rejected requests and 135 messages that have not been acknowledged as delivered. It corresponds respectively to 2.45% and 0.93% of the total amount of messages processed. The results obtained could be even better, since we reported a large concentration of failed requests at the end of the last test case, due to quota limits exceeded.

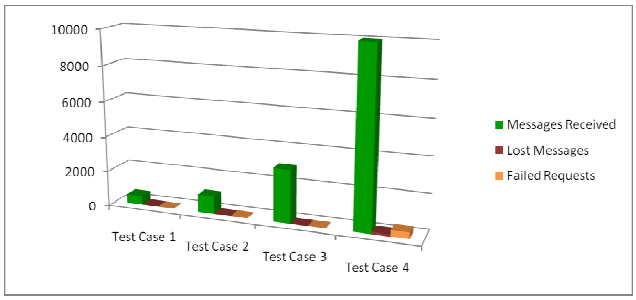


Figure 7. Message Delivery 3D Grouped Column Chart

In the same stress test scenario, we collected a set of performance related metrics, as shown in Table 1. The middleware comfortably supported the intense load of requests, maintaining good performance levels. Based on the results obtained, a few observations can be made about this platform. Throughput and performance are positively impacted under heavier load of concurrent requests. We believe that it happens due to reconfiguration algorithms implemented by the cloud platform, scaling to uphold the increasing volume of requests. However, we have also observed that the reliability of message delivery is negatively impacted under these circumstances.

TABLE I. PERFORMANCE RELATED METRICS

Number of Samples	Median (ms)	Throughput	kB/sec	Lost Messages	Failed Requests
500	691	48,4/sec	474,75	0	0
1000	394	55,2/sec	949,5	0	0
3000	293	103,4/sec	2848,5	3	0
10000	204	177,7/sec	9495,0	132	356

In the test scenario executed at a registered callback endpoint environment, we collected and measured the total amount of time for every notification to be delivered to all of the subscribed endpoints, as described on the previous subsection. Figure 8 presents the total message delivery time measured at specific intervals during the test execution. The x-Axis represents time (HH:MM:SS) intervals at which messages were delivered, while the y-Axis represents the total message delivery time in milliseconds for each notification sent during the test.

 ISBN: 978-1-61208-438-1

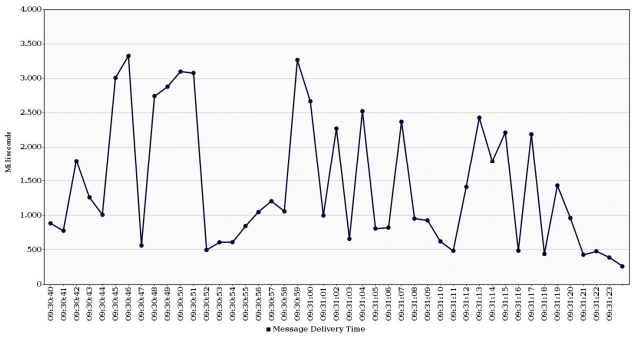


Figure 8. Total Message Delivery Time

Based on a string of experiments, we conclude that the differences observed in the results, exhibited on Figure 8, occur as a consequence of the following events: Network jittering, resource availability, and concurrency. The simulation results show that the performance provided by the platform, in terms of delivery ratio, throughput and timeliness is suitable to the middleware purpose. This is a consequence of the use of an asynchronous messaging approach and a cloud-based platform, capable of scaling and performing according to the restrictions imposed. Therefore, it is possible to offer a high degree of scalability using a publish/subscribe middleware for real-time disease surveillance. Even for larger applications scenarios, where the number of subscribed healthcare applications is considerably higher, or where the messages exchanged are in a higher number, we believe that it is possible to scale in order to support the increase of load. Further experiments have shown that when the number of concurrent messages is increased, the middleware's throughput is higher, while the performance is stable. However, with a higher amount of messages exchanged, we have observed an increase in the number of lost or unacknowledged messages.

V. RELATED WORK

Much of the related work has been covered in the previous Sections of this paper. In this Section, we revisit some of the topics discussed and present them in a summarized view.

Healthcare Interoperability. As mentioned on Section 2, subsection B, multiple approaches have been attempted in order to promote interoperability initiatives for healthcare systems, as demonstrated on both [3] and [24]. Recent work has been developed in order to offer a powerful and extensible standard specification for healthcare system-level integration, namely FHIR [9]. In our work, we have slightly extended the FHIR data model in order to include statistical information to be further processed by a CEP unit.

Disease Surveillance. This is an emerging field of research that has been achieving significant success in the early detection and report of disease outbreaks at regional scale. The DHIS 2 project as described in [1] uses Java enabled phones to send health related data using SMS. This is an example of how a low-cost disease surveillance mechanism can be helpful in the prevention or mitigation of occurrences, especially in developing countries [2]. Another successful example can be found in India and Sri Lanka, as described in the work of Waidyanatha *et al* [4]. The T-Cube project has been developed in order to detect emerging patterns of potentially epidemic events based on the analysis of digitized clinical health records. On our work, we use a similar approach, as previously described.

The ESS project developed in Sweden is an Event-based Surveillance System that uses records of telephone calls to the Swedish National Health Service, in order to monitor unusual patterns [25]. Statistical analysis is performed over collected data to calculate deviation limits. In an attempt to process larger datasets, Santos and Bernardino presented a system architecture for near real-time detection of epidemic outbreak at global scale using on-line analytical processing (OLAP) techniques [26].

VI. CONCLUSIONS

This paper has presented a middleware platform responsible for receiving and interpreting data informed by healthcare organizations, and based on the results obtained through data analysis, the middleware publishes real-time notifications to all healthcare applications subscribed to this platform.

There is an increasing need for timely delivery of messages and notifications, in very large user base platforms. In this context, it is extremely important to develop a platform capable of scaling to sustain the expected levels of performance and throughput under growing demand. PREVENT uses the FHIR specification in order to exchange system-level messages, presenting a market-friendly environment for real-time integration of applications. FHIR is currently published as a Draft Standard for Trial Use (DSTU), hence we hope that this work may serve as a contribution on the promotion of interoperability initiatives, and a step towards the development of an international disease surveillance platform.

We believe that several factors combined make PREVENT a scalable and efficient platform:

- An asynchronous event-based approach for message processing, reducing network contention and the number of threads needed to process the same workload;
- A network-efficient non-blocking I/O communication model for HTTP connections;
- And a cloud hosting infrastructure.

The results obtained from our experiments demonstrate that a cloud-based platform using the publish/subscribe pattern for real-time notifications represents an appropriate choice, in order to assure time-constrained delivery of mission-critical data.

The contribution of this paper is a first step to enabling the use of the FHIR specification for healthcare system integration in order to support a global system-level

 ISBN: 978-1-61208-438-1

outbreak warning platform. Our ongoing research aims to perform experiments using heterogeneous environments and datasets, in order to present richer interoperable scenarios.

As future works, we plan to implement several extensions on our middleware platform in order to support functionalities like: security using OAuth; big data analytics for prediction using information extracted from notifications received; support for legacy and multi-format messages (HL7 v2, short message services, etc.) using a message adapter layer for data transformation, and multiprotocol integration using TCP, HTTP and HTTPS.

REFERENCES

[1] L. Pascoe, J. Lungo, J. Kaasbøll, and I. Koleleni, "Collecting Integrated Disease Surveillance and Response Data through Mobile Phones," IST-Africa Conference Proceedings, 2012, pp. 1-6.

[2] C. Déglise, L.S. Suggs, and P. Odermatt, "SMS for disease control in developing countries: a systematic review of mobile health applications," Journal of Telemedicine and Telecare, 2012, v. 18, n.5, pp. 273-281.

[3] T. Tran, H-S. Kim, and H. Cho, "A Development of HL7 Middleware for Medical Device Communication," Proc. 5th ACIS Int. Conf. Softw. Eng. Res., Manage. Appl., 2007, pp. 485-492.

[4] N. Waidyanatha et al., "T-Cube Web Interface as a tool for detecting disease outbreaks in real-time: A pilot in India and Sri Lanka," IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010, pp. 1-4, doi: 10.1109/RIVF.2010.5633019.

[5] "Constitution of the World Health Organization", 2005. [Online]. Available: http://goo.gl/2vvJBS. [Accessed: 30-May-2015].

[6] "International Statistical Classification of Diseases and Related Problems", 2010. [Online]. Available: http://goo.gl/FCp5Js. [Accessed: 30-May-2015].

[7] "International Health Regulations", 2005 Second Edition. [Online]. Available: http://goo.gl/m8Rajk. [Accessed: 30-May-2015].

[8] "Health Level Seven International – Homepage", 2015. [Online]. Available: http://www.hl7.org/. [Accessed: 31-May-2015].

[9] D. Bender and K. Sartipi, "HL7 FHIR: An Agile and RESTful approach to healthcare information Exchange," Computer-Based Medical Systems (CBMS), IEEE 26th International Symposium, 2013, pp. 326-331.

[10] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, University of California, Irvine, 2000.

[11] T. Velte, A. Velte, and R. Elsenpeter, "Cloud computing, a practical approach." McGraw-Hill, Inc., 2009.

[12] P. Jogakekar and M. Woodside, "Evaluating the scalability of distributed systems," IEEE Transactions on Parallel and Distributed Systems, 2000, v. 11, n. 6, pp. 589-603.

[13] D.F. Garcia, R. Garcia, J. Entrialgo, J. Garcia, and M. Garcia, "Experimental evaluation of horizontal and vertical scalability of cluster-based application servers for transactional workloads," 8th International Conference on Applied Informatics and Communications, 2008, pp. 29-34.

[14] D. C. Luckham and B. Frasca, "Complex event processing in distributed systems," Computer Systems Laboratory Technical Report CSL-TR-98-754, Stanford University, Stanford, v. 28, 1998.

[15] V. Vaidehi, R. Bhargavi, K. Ganapathy, and C.S. Hemalatha, "Multi-sensor based in-home health monitoring using Complex Event Processing," International Conference on Recent Trends In Information Technology (ICRTIT), 2012, pp. 570-575.

[16] D. Robins, "Complex event processing," Second International Workshop on Education Technology and Computer Science, Wuhan, 2010. [Online]. Available: http://goo.gl/jLROpc . [Accessed: 30-May-2015].

[17] "Google App Engine". [Online]. Available: https://cloud.google.com/appengine/docs/. [Accessed: 13-June-2015].

[18] "Google Cloud Pub/Sub". [Online]. Available: https://cloud.google.com/pubsub/docs/. [Accessed: 13-June-2015].

[19] "FHIR Specification Home Page- FHIR v. 0.0.82", 2015. [Online]. Available: http://goo.gl/RDwuPm. [Accessed: 31-May-2015].

[20] "Apache HttpComponents." [Online]. Available: https://hc.apache.org/. [Accessed: 13-June-2015].

[21] L. Lamport, "The part-time parliament," ACM Transactions on Computer Systems (TOCS) Journal, 1998, v. 16, n. 2, pp. 133-169.

[22] "Digital Ocean" [Online]. Available: https://www.digitalocean.com/. [Accessed: 13-September-2015].

[23] Byung-Jae Kwak, Nah-Oak Song, and Miller, M.E., "Analysis of the stability and performance of exponential backoff," IEEE Wireless Communications and Networking, 2003, v.3, pp. 1754-1759, doi: 10.1109/WCNC.2003.1200652.

[24] Fabio Vitali, Alessandro Amoroso, Marco Rocetti, and Gustavo Marfia, "RESTful Services for an Innovative E-Health Infrastructure: A Real Case Study," IEEE 16th International Conference on e-Health Networking, Applications and Services, 2014, pp. 188-193.

[25] Deleer Barazanji and Pär Bjelkmar, "System for Surveillance and Investigation of Disease Outbreaks," 23rd International Conference on World Wide Web Pages, 2014, pp. 667-668.

[26] Ricardo Jorge Santos and Jorge Bernardino, "Global Epidemiological Outbreak Surveillance System Architecture," 10th International Database Engineering and Applications Symposium, 2006, pp. 281-284.

 ISBN: 978-1-61208-438-1

Requirement's Variability in Model Generation from a Standard Document in Natural Language

Juliana Galvani Greghi*†, Eliane Martins†, Ariadne M. B. R. Carvalho†
*Department of Computer Science
University of Lavras
Lavras-MG Brazil
Email:juliana@dcc.ufla.br
†Institute of Computing
University of Campinas
Campinas-SP Brazil
Email: {eliane, ariadne}@ic.unicamp.br

***Abstract*—Requirement documents are, in general, written in natural language and, therefore, may contain problems such as ambiguities and inconsistencies. In some domains, such as the aerospace domain, requirements are obtained from standard documents that describe the set of services to be implemented. The standard document may present variability, with a set of mandatory requirements, which must be always implemented, and a set of optional ones. Also, different applications may require different sets of services. In general, models, such as Extended Finite State Machines (EFSMs), are used to represent the requirements. Because of the variability found in the document, different models can be generated, making manual modeling a difficult and error-prone task. There are many approaches to (semi) automatic generation of models from requirement documents. But, to the best of our knowledge, none is applied to standard documents, or considers variability issues. In this work, a method for semi-automatic generation of EFSMs from a natural language standard document, with the use of variability modeling, is presented. To validate the proposed approach, a prototype to generate different EFSMs for the same service was developed, considering the commonalities and variabilities of the requirements. The document describes services and protocols for applications in the aerospace domain.**

Keywords–requirements modeling; variability management; aerospace domain.

I. INTRODUCTION

The use of models in Software Engineering is not new, because they are less subject to ambiguities and inconsistencies than natural language descriptions. Besides, models can be automatically processed by tools, enabling not only their validation, but also the derivation of code, as well as test cases. However, modeling is a manual task, error-prone [1] due to human interference. Many applications may show some degree of variability, and the definition of commonalities and variabilities enables reuse of software artifacts, ensuring the quality of the developed applications [2]. Reuse decreases development and testing time, and increases reliability of the reused elements, as much as these artifacts will have been validated in previous applications [3].

Many different models may be required in the presence of variability, catering for mandatory requirements, and for many possible combinations of mandatory and optional requirements. In this context, the following research questions must be answered: Can we deal with the large number of different models that may be generated, considering the combination of mandatory and optional requirements? Can these models be semi-automatically generated?

Some approaches for automatic conversion of texts into models [4][5][6] are detailed in Section III-A. But, to the best of our knowledge, the available approaches do not address variability in the requirements, i.e., when optional requirements make it possible to create different behavioral instances, with the same core asset, which allow the final product to have different capabilities.

In this work, product line engineering concepts were applied to a standard document to deal with the variability of the requirements, and to be able to generate models that represent the services described in the standard document. The analyzed standard document was elaborated by the European Cooperation for Space Standardization (ECSS). The document is the *ECSS-E-70-41A - Ground System and Operations - Telemetry and Telecommand Packet Utilization*, called Packet Utilization Standard (PUS), which standardizes telecommand and telemetry packets related to a set of services for handling on-board data software [7]. Those requirements are applicable to every space mission, and they take into account that different missions require different functionalities [7]. Furthermore, variability modeling of the services described in the PUS document helps the expert user, presenting the features that must be included in the implementation, as well as the optional ones, in a self-contained, summarized, and graphical format. Moreover, the model shows relations between features, suggesting features that should be made available together, and the ones that should not, thus reducing the human effort required to identify the information needed to implement a service.

The objective of this work is to show how to use variability modeling to represent functional requirements from standard documents, and to semi-automatically generate models representing the requirements of each service described in the standard document.

It is important to say that the EFSMs [8] are generated from the standard document, not from the variability model. The variability model is used to help understanding the relationship between the services and their functionalities, which are described in the standard document, and in the development

 ISBN: 978-1-61208-438-1

of a product line to semi-automatically generate EFSMs representing the requirements defined in a standard document. The proposed approach was evaluated with a prototype to semi-automatically generate EFSMs from standard documents. Furthermore, an empirical work was conducted in which the PUS document [7] was used. For that, we had the collaboration of researchers from the National Institute for Space Research (INPE), and from the Aeronautics Technology Institute (ITA), who provided the requirements for prototype development, and who informally evaluated the generated models. They are very familiar with the PUS document [7] because it is used in the development of some of their projects.

The rest of the article is organized as follows. Section II introduces the main concepts related to software product line, and variability management. Section III presents related work, and Section IV introduces the proposed approach. Section V shows a working example, and Section VI presents the validation of the proposed feature model. Finally, Section VII concludes the article with suggestions for further work.

II. Background

Software Product Line (SPL) is a set of software systems that share common and optional features. It specifies particular needs, obtained from a set of common assets [9]. Software Product Line Engineering (SPLE) is a paradigm for developing software applications using a common structure and satisfying individual client needs. These individual needs are the variability of the system [2].

There are different definitions for variability. In this work, the following definition is used: "Variability is the ability to change or customize a system" [10]. There are several methods for variability modeling, many of them based on features [11]. The concept of feature modeling was first proposed within the Feature-Oriented Domain Analysis (FODA) method [12]. A feature is an important quality or characteristic of a software system, visible to the user. Feature Model (FM) is used to express common and variable system requirements in a certain domain, and the relationship between them [12]. FM is used for product line development, and it represents the properties of the system that are relevant to the end users in a hierarchical form [11].

Feature modeling was chosen because it can be easily understood by stakeholders, and it is widely accepted by software engineers [11]. The notation used, presented in [13] *apud* [14], extends the feature model notation with the feature type *or-feature*, improving the representation of the relationship between features.

III. Related Work

There are many approaches for model generation from natural language documents. The most relevants to this work are presented in Subsection III-A. The application of SPL in the space domain has been encouraged along the years, and some examples of these applications are presented in Subsection III-B.

A. Model Generation from Natural Language Information

The transformation of natural language requirements into models present many challenges. Sometimes documents have to be rewritten, either to correct mistakes, or to express information in a structured format. Moreover, the notation in which the information is expressed, and the kind of model that must be generated, should be taken into account.

The approaches presented here use some sort of Natural Language Processing (NLP) technique for text pre-processing, usually a syntactic analyzer (or parser), and a part-of-speech (PoS) tagger. The parser provides a valid syntactic structure for a sentence. The process is supported by a grammar that defines a set of valid structures in a given domain [15][16]. The PoS tagger associates syntactic and morphological features to the words of a sentence or text [15].

Yue, Briand and Labiche [17][4] present a method, implemented as part of the aToucan tool, which takes use cases in a pre-defined format [18] as input, extracting information and producing Unified Modeling Language (UML) models (class, activity, and state diagrams). Information extraction is performed by NLP tools and transformation algorithms. An extension of this work is presented in [19] to generate Aspect-based State Machines. A product line model and configuration were proposed in [20] to support model-based tests.

Deeptimahanti and Babar [21] transform the requirements document into UML models (class, use cases, and collaboration diagrams), using a tool called UMGAR. The process initiates with requirements being rewritten to remove eventual ambiguities from the sentences. Then, the information is extracted to generate the analysis model. The model can be converted to Extensible Markup Language Metadata Interchange (XMI) files, allowing for requirements traceability [5].

Kof [6] presents an approach to convert textual descriptions of an automaton into a finite automaton model. The text is pre-processed using a PoS tagger. The identification of states relies on search terms, defined by the user, and on specific word categories. The identification of transitions is based on sentence's segmentation, and on the categorization of the sentence segments. A set of rules was defined to extract the text segments that should be used for model generation. The generated automaton is represented in a tabular form.

Kof and Penzenstadler [22] present a method for integration of an NLP approach and a CASE tool. They show a methodology for translation from text to model using a previous work [6]. The training data requires that the user selects the section of the document to be processed. The methodology was integrated with a tool that enables user interaction. The tool learns on the fly about words and grammar constructions that can be used in model generation.

Santiago [23] presents a methodology, SOLIMVA, to generate model-based test cases from requirements in natural language documents. A statechart is produced with the information extracted from the requirements and a tool, called GTSC [24], is used to generate test cases.

In our approach, the input is not restricted to a specific format, as in [4][17], and sentences are not rewritten, differently from [21][5]. The search for patterns is similar to [6], but in addition, our approach uses patterns extracted from the document's structure to obtain information that is used for model generation. Similarly to Santiago's work [23], our approach was applied to the aerospace domain. All of them use NLP tools to extract information from a natural language document.

The novelty of the proposed approach is variability handling: to the best of our knowledge, none of the existing approaches

 ISBN: 978-1-61208-438-1

deals with requirements variability during model generation. In our approach, commonalities and variabilities are identified and modeled with an FM, and this information is used to guide model generation.

The information extraction process is briefly described in Section IV.

B. Product Line for Space Application

The use of SPL approaches in space applications has increased along the years due to many factors, e.g., search for more rigorous architectural definition, cost saving, and the need for more systematic approaches for systems development [25][26][27][28].

Many approaches apply product line concepts in the aerospace domain and, in general, the final products are specific software applications [28][29]. Our approach, differently, applies product line concepts for model generation.

IV. Representing variability for EFSM model generation

The goal of this work was to use variability modeling to help the semi-automatic generation of EFSMs. Variabilities were identified to facilitate the generation, from standard documents, of models used for development and testing. To exemplify the entire process, a sample text, with some characteristics often found in standard documents, is presented in Figure 1.

This is a made up data packet validation protocol, illustrating the proposed information extraction methodology.
Standard document for network communication protocol
Chapter 3 - Data packet validation
3-A Description
This chapter defines a protocol for data packet validation. Data is organized in packets, as described in chapter 1, and these packets must be validated before sending. When a packet is sent, a validation activity is initiated. The mandatory fields must be filled always, on each data transfer. Optional fields can be filled according to the application requirements.
3-B Concepts
Data packet validation occurs in two stages:
1) Verification of mandatory fields: the type and the size of data must be verified. Data must conform to the allowed types, and the size must comply with the restrictions of the used type. An error shall occur if any mandatory field is not filled.
2) Verification of optional fields: if one (or more) optional field is filled, the data value must be verified. Data must conform to the allowed values for the field. If a field value is out of range, a failure report must be generated.
3-C Available functionalities
3-C.1 Basic functionalities
Verification of mandatory fields (1,1): the result of the verification must be recorded in a report file.
Verification of optional fields (2,1): the result of the verification can be recorded in a report file if it is required by the application.
In addition to the description, other details about the protocol are described in the last chapters of this manual. For details consult Appendix A.
3-C.2 Additional functionalities
Sending a warning e-mail (3,2)
Registering a log file for validation errors (4,2)
Validation of data packet integrity (5,2)
3-D Dependencies

ID	Name	Requires
1	Verification of mandatory fields	
2	Verification of optional fields	
3	Sending a warning e-mail	
4	Registering a log file for validation errors	
5	Validation of data packet integrity	1,2

Figure 1. Text fragment to illustrate the proposed methodology.

A. EFSM model generation

Figure 2 shows the process diagram for State Machine Generation. The entire process has seven steps, subdivided in sixteen tasks, which are described next. The process of information identification and information extraction was already discussed in a previous work [30].

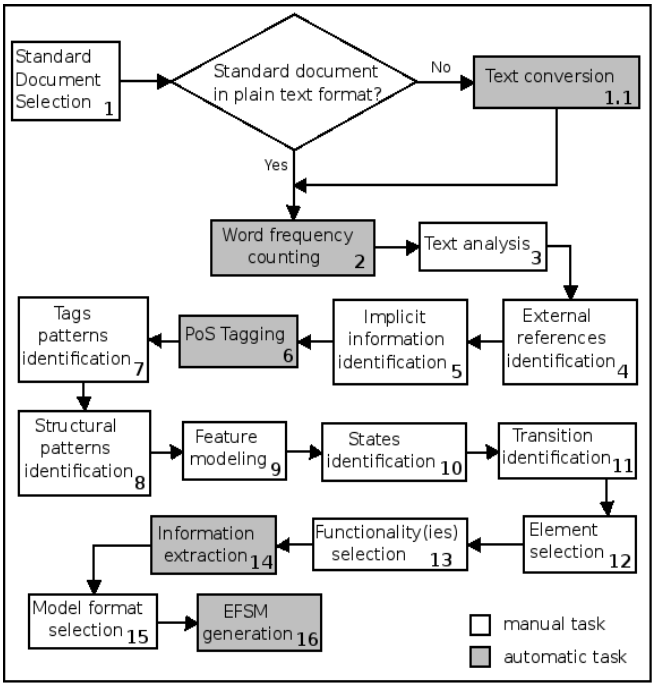


Figure 2. Process Diagram for State Machine Generation.

Step 1 - Semi-automated analysis of the standard document

This step comprises a set of activities performed by an expert analyst. The process initiates with the selection of the standard document to be processed (Task 1 - Figure 2). The selected document must be in plain text format (.txt) to be processed by the NLP tools. If the text is available in a different format, it must be converted to .txt (Step 1.1 - Figure 2). The plain text standard document is then processed by a word frequency counter tool, and the results are used in the analysis of the text (Tasks 2 and 3 - Figure 2). This analysis helps to identify structural patterns in the sentences. Stop words, i.e., words that have a high frequency but no semantic meaning [31], are not taken into account in the analysis. An example of an identified pattern is: most sentences initiating with the word "if" present a condition for the execution of an action, as it can be noticed in the sentence "If a field value is out of range, a failure report must be generated.", presented in Section 3-B.2, Figure 1.

Step 2 - Reading the text

After finding patterns in the sentences, the next task is the careful manual reading of the standard document to identify references to other documents, or to a different section or chapter of the same document (Task 4 - Figure 2). This information is marked for future processing.

The next task is to identify implicit information, i.e, information represented in tables or figures that are not explicit in the text, and that are related to the described service (Task 5 - Figure 2). This information must also be marked and processed to be used in a later stage. In the proposed methodology, information described in different chapters/sections, or as

figures or tables, is manually extracted and stored in a database for EFSM generation. This fact can be observed in Section 3-D, line 5, Figure 1: the dependency relation is not explicit in the text, and must be manually extracted and stored in a database to be used during model generation.

Step 3 - Reading the service description

Text is pre-processed using the Stanford PoS Tagger [32] (Task 6 - Figure 2). Each category attributed to a word is a tag that can be used to help the expert analyst in the extraction process. Sequences of tags were already used by Kof [6], and his research is the work mostly related to ours. In Figure 3, we can observe sequences of tags. In the underlined text segments, the sequence "noun (tags initiated by NN) preposition (tag IN) adjectives (tag JJ) noun (tags initiated by NN) noun (tags initiated by NN)" is recurrent and can be used to identify sets of information that must be extracted and used in later stages. The process of finding patterns in the sequences of tags is represented, in Figure 2, by Task 7.

In addition to the categories, the structural pattern identified in the document can also be used during information extraction. The identification of structural patterns is represented, in Figure 2, by Task 8. An example of a structural pattern can be found in Section 3-C.1, Figure 1. There is a description of a functionality, followed by a parenthesis, two numbers separated by a comma, and another parenthesis. The first number is an identification of the functionality, and the second indicates if it is a mandatory functionality (number 1), or an optional functionality (number 2). The pattern "description parenthesis number comma number parenthesis" can be used in the extraction process to help the identification of information. After the analysis, the expert analyst must identify mandatory and optional functionalities described in the text. These functionalities must be represented in a feature model, which is used to guide the selection of services and functionalities in a later stage of the processing (Task 9 - Figure 2).

Step 4 - Identification of states and transitions

The process continues with the identification of states (Task 10 - Figure 2), which relies on the location of the information, e.g., on the section of the document where the information is described, and on the sequence of the tags. In Figure 1, we can see that states are located in the "Concepts" section, identified by the number of the chapter, and by the identification of the section (3-B). We can also observe that the description of the stages starts with a noun (tags initiated with NN), followed by "of" plus and adjective, and finished with a noun. This pattern may be identified in Figure 3, in the underlined text segments. The tasks to identify transitions are very similar to those for state identification, and represented, in Figure 2, by Task 11. A transition is composed of events, actions and conditions, also known as guards. The identification of each transition component can follow a set of rules and patterns.

The identification of conditions and actions, for example, is based on the keyword "if". In Figure 1, we can observe two different situations: the if-clause at the beginning of the sentence (Section 3-B.2, Figure 1 - "If a field value is out of range, a failure report must be generated."), and the if-clause at the end of the sentence (Section 3-B.1, Figure 1 - " An error shall occur if any mandatory field is not filled."). In the first situation, the text segment between the conjunction and the comma is a condition ("a field value is out of range"), and the text segment after the comma is an action ("failure report must be generated"). In the second situation, the action is the text segment until the conjunction ("An error shall occur") and the condition is the text segment after the conjunction ("any mandatory field is not filled."). Patterns are used to create sets of rules to extract information from the standard document. The set of rules must cover the entire description of a service present in the standard document. It is possible that each service may need a specific set of rules, because the methodology is highly dependent on the text structure and on the writing style.

a_DT -RRB-_-RRB-Verification_NNP of_IN mandatory_JJ fields_NNS :_: the_DT type_NN and_CC the_DT size_NN of_IN data/NNS (...)
b_DT -RRB-_-RRB- Verification_NNP of_IN optional_JJ fields_NNS :_: if_IN one_CD -LRB--LRB- or_CC more_JJR -RRB--RRB- optional_VBD (...)

Figure 3. Text segments for tag pattern identification

Step 5 - Selection of the service and functionalities to be modeled

This activity must be performed by an expert user, who is using the requirements described in the standard document for the development of a new system. The user must select the service that needs to be modeled (Task 12 - Figure 2). The mandatory functionalities are selected by default, but the expert user may select the optional functionalities that must be present in the model (Task 13 - Figure 2). The information about what is mandatory or not in a service is provided by the feature models elaborated in Task 9, Figure 2.

Step 6 - Information extraction

The information extraction is an automated task (Task 14 - Figure 2), which uses the sets of rules created during Step 4. The rules are applied in the standard document to extract information about a specific service, selected by the expert user in Task 12, Figure 2. The entire information about the selected service is recovered. After the extraction process is completed, the necessary information to represent only the functionalities selected in Task 13, Figure 2, is filtered and sent to the next step, that is, generation of EFSM. The process of information filtering is based on mandatory and optional features identified through the feature models elaborated in Task 9, Figure 2.

Step 7 - Generation of the EFSM

In the last step of the process, the expert user must select the model format (Task 15 - Figure 2). The text format is generated by default, but the user may request the generation in graphical format too. The last activity is the generation of the model (Task 16 - Figure 2). The extracted information filtered according to the selection made by the expert user is used to generate the EFSM.

B. Variability *Modeling*

As mentioned in Section II, variability is expressed through a Feature Model (FM), following the FODA method [12]. The three steps for feature modeling are described next.

Feature Identification

A feature is an aspect or characteristic of the system considered important by its users [12]. In our proposal, features are defined according to the expert user's viewpoint.

The user may select the standard document version, the text converter and the PoS tagger. The text converter is used to pre-process the standard document, and the PoS tagger is used for annotation. These requirements are considered features because they are directly related to the choices of the expert user that can affect the configuration of the new model.

Regarding the state machine model, the user must select the service(s) to be modeled, the information about the EFSM that will guide model generation, and the output notation of the model. These are the main mandatory features of the system. The sub-features identified for each of these main features are as follows.

Feature modeling

According to the FODA method [12], feature commonalities and variabilities are represented as a tree. Figure 4 presents the feature diagram for the translation system *txt2smm*.

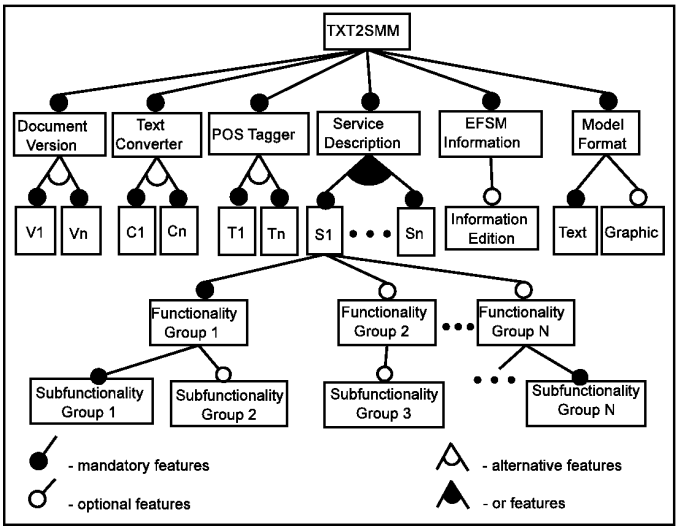


Figure 4. Feature diagram for the txt2smm system.

The generation of a new model requires the configuration of six main mandatory features. Three of them are related to document management: "Document Version", "Text Converter" and "PoS Tagger", each with alternative sub-features. In an alternative relationship, only one sub-feature must be selected.

"Document Version" presents the sub-features "V1" and "Vn", representing that the user must select the version of the standard document. "Text Converter" presents the sub-features "C1" and "Cn", representing that the user must select the converter tool. Finally "PoS Tagger" presents the sub-features "T1" and "Tn", representing the user must select the PoS Tagger to be used for text annotation.

"Service Description", "EFSM Information" and "Model Format" are the next three features, related to the state machine generation. "Service Description" presents the *or-features* "S1" and "Sn", which means that at least one service must be selected. Each service has a set of mandatory functionalities, always available in every implementation of the service, and which are represented by the mandatory feature "Functionality Group 1". Additional functionalities are represented by optional features "Functionality Group N". Each Functionality may describe a set of sub-functionalities such as reports, messages, or requests.

These are represented by "Sub-functionality 1" to "Sub-functionality N", which may be mandatory, optional, alternative, or or-feature, depending on the requirements of each functionality.

The information extracted for state machine generation is presented to the expert user who may edit it. This is represented by the mandatory feature "EFSM Information", and by the optional feature "Information Edition". The state machine model can be generated in textual or graphical notation. This selection is represented by the mandatory feature "Model Format", and the available formats are represented by the mandatory feature "Text", and by the optional feature "Graphic".

Applying feature modeling to the text example in Figure 1, and considering that "Data packet validation protocol" is one of the protocols described in the standard document, the resultant FM would look like the diagram shown in Figure 5, where "Data transfer protocol" is represented by "S1".

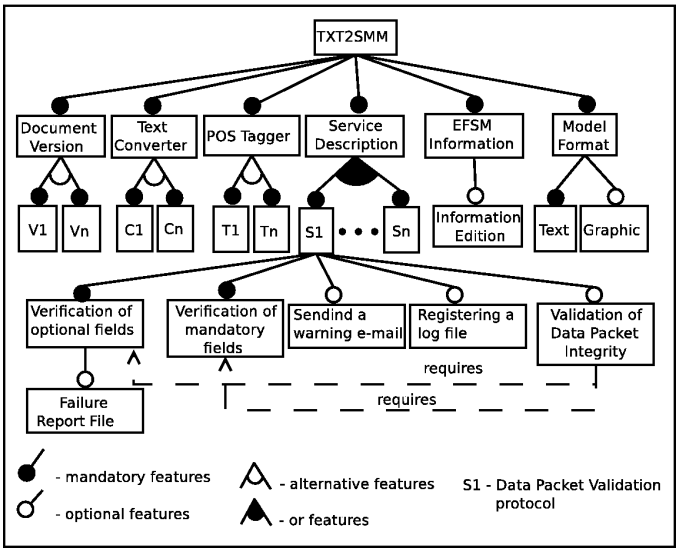


Figure 5. Feature diagram for the text example.

The service described in the text example is a made up data packet validation protocol, and the available functionalities are: Verification of mandatory fields, Verification of optional fields, Sending a warning e-mail, Registering a log file for validation errors, and Validation of data packet integrity. The first two functionalities are mandatory, i.e., they must be always available in every implementation of the protocol, and the last three functionalities are optional. The text also describes the generation of a report file for "Verification of optional fields", which is mandatory. The dependency relation is described in Section 3-D, Figure 1.

Feature model validation and analysis

There are many tools for supporting feature modeling, but some are more adequate for feature checking; to validate and analyze the generated model, one, to be presented later, was chosen. Several operations for automated analysis of feature models are available, like: validation of the feature model, i.e., whether the FM allows for the configuration of at least one valid product; estimation of the number of valid products; computation of all valid products; evaluation of the variability degree of the feature model; checking the consistency of the product, i.e., determining if a specific combination of features is valid, looking for errors in the feature model, and many other operations. The feature model validation process is presented in Section VI.

V. WORKING EXAMPLE

The proposed approach was applied to the PUS document [7]. A prototype was developed to extract the necessary information, and to generate different models for the same service, taking into account the commonalities and variabilities of the service. The text was pre-processed using the Stanford PoS Tagger [32].

The PUS document describes a set of 16 services to support functionalities for spacecraft monitoring, but there are no mandatory services for a given mission. Each service is described with a minimum capability set that must be always included in every implementation of the service, and additional capability sets that may be optionally implemented [7]. In addition to the variability, the document also defines

 ISBN: 978-1-61208-438-1

other important requirements that must be considered in the implementation of each service, and which may appear in different sections of the PUS document.

An example of an instantiation of the feature model for the generation of a state machine of the Telecommand Verification Service (TVS) is shown. This example was executed using the prototype especially developed for the task.

The TVS is described in four stages, each one with a set of capabilities that might be mandatory or optional. Each capability defines the generation of reports describing the failure or the success of a given stage. According to the chapter that describes the information that can be used for any service, failure reports must always be generated and, according to the description of the service, there will be a dependency relationship between success and failure reports [7].

Regarding document processing, the following features were selected: standard document version: V1, corresponding to the version from 30 January, 2003; Text converter: C1; POS Tagger: T1; Service: TVS; Minimum Capability: Capability Group 1, corresponding to Acceptance of the telecommand; Additional Capabilities: Capability Group 4, corresponding to the Completion of the Execution of the telecommand; Edit Information: No edition; Model Format: text.

With this configuration, an EFSM in text format is generated, and shown here in a tabular form (Table I). The first column shows the state names, extracted from the PUS document. The state "withoutTC" is not presented in the text, but it was included in the EFSM as recommended by the researchers from INPE and ITA who informally evaluated the generated models.

The second and third columns are names used to identify input and output events for each state. These names were semi-automatically extracted from the document during the information extraction phase, presented in Figure 2. This simple example shows the state machines generated to represent the normal behavior of the Telecommand Verification Service. The resultant model can be seen in Figure 6.

The model presented in Figure 6 represents a semi-automatically generated EFSM for the example presented in Table I. It was redrawn, and the natural language conditions were suppressed due to the to lack of space. In this model, the service is represented in two main states: the acceptance of the telecommand, and the completion of the execution of the telecommand. For each of these states, there is an available report. In Figure 6, this report is represented by a natural language parameter, followed by the terms AckON or AckOFF. These terms refer to the possibility of sending or not the report to the control station.

To evaluate the generated model, the model presented in Figure 6 was compared to the EFSM manually generated (Figure 7), available at [33].

These models represent the same capabilities configured in the example (Table I), and some differences between the models were found: an additional state present in Figure 7, which is represented in the semi-automatically generated EFSM as a natural language condition; the information about sending reports to the control station that is represented, in Figure 7, by brackets and hifens. Despite of these differences, the main information were preserved, and the semi-automatically generated model can be used by the expert user as a initial model for the development and testing processes.

This working example helps to answer the first research question presented in Section I: Can we manage the large number of different models that can be generated, considering the combination of mandatory and optional requirements? The product line approach helps to deal with the different models which may be generated for the same service, considering the possible combinations between mandatory and optional features. The implementation of the prototype and the evaluation of the generated models helps to answer the last research question presented in Section I: the models can be semi-automatically generated, and the possible combinations are generated in a controlled way, avoiding invalid feature combination.

TABLE I. OUTPUT FROM THE STATE MACHINE GENERATION SYSTEM

State	Input Event	Output Event
without TC	TC	TC
Acceptance of the telecommand	TC	Acceptance OK
Completion of execution	Acceptance OK	Completion OK

VI. FEATURE MODEL VALIDATION

The feature model presented in Figure 4 was validated within the FaMa FW [34] tool, which represents variability via feature models. FaMa FW performs many analysis operations on an FM, for example, checking if the FM has at least one valid product, computing all valid products for a given FM, and the variability degree of the FM.

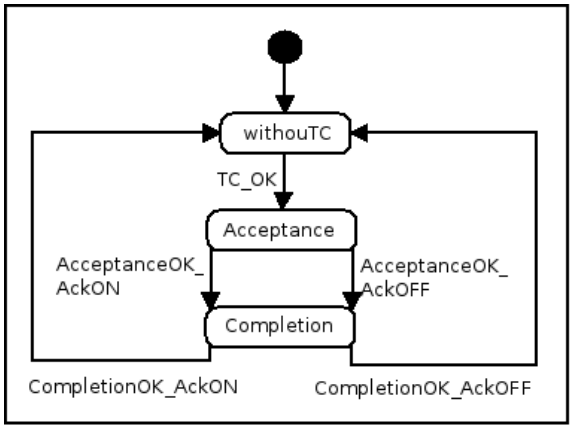


Figure 6. EFSM semi-automatically generated.

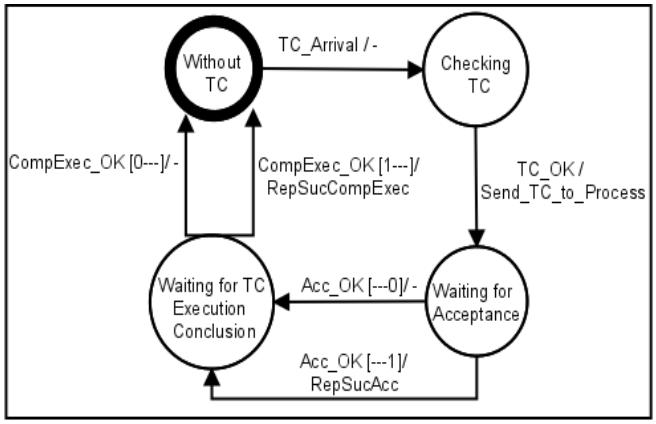


Figure 7. EFSM manually generated. Adapted from [33]

FaMa FW can be used in a shell front-end, or integrated to other applications. In this work, the shell front-end was used. All available operations in the command line interface were applied, and no errors were found.

For instance, the TVS affords two document versions, two converter tools, two PoS taggers, the different combinations of capabilities from TVS, the possibility of editing information, and two model notations to represent the EFSM. Choices between the several alternatives multiply to give 256 descriptions of new products. The number gets even larger if it is taken into account that the PUS document describes 16 services.

The prototype avoids the selection of invalid combinations, for example, automatically selecting mandatory features from a selected service by the expert user, or automatically selecting/deselecting features related to an optional, or alternative feature selected by the expert user.

Regarding the selection used in the working example for the Document Version, the PoS tagger, the text converter and the selected service, there are 8 different possible combinations of additional capabilities for the TVS.

Some limitations were identified in the proposed approach: self-loops and hierarchies cannot be treated unless explicitly described in the text.

VII. Conclusion and Future Work

The use of models to represent requirements is widespread as an alternative to minimize problems during development and testing processes of computational systems. In an attempt to help the semi-automatic generation of EFSMs, considering the variability of requirements in a natural language standard document, the following questions were posed in Section I: Can we deal with the large number of different models that may be generated considering the combination of mandatory and optional requirements? Can these models be semi-automatically generated?

These questions were answered in the article, showing that variability modeling is fundamental to the successful generation of state machine models, not only by allowing for the exploration of numerous combinations, but also for avoiding the invalid ones.

A prototype tool was implemented to semi-automatically generate state machine models. The models were generated according to the proposed feature model, and informally evaluated by the INPE and ITA researchers, who also selected the set of services to be modeled. The state machine models generated with the prototype tool were validated using a model checking tool, and by comparing the semi-automatically and manually generated models for the same services, with the same configuration.

Results showed that the state machines are very similar. The similarities are a strong indication that the generated models could help the analysts in the development and testing processes of new computational systems, reducing time and effort needed for manual modeling.

Although the effort needed to extract information from the document may seem excessive due to the writing style dependency of the standard document, this can be counterbalanced by several aspects: reduction of time consumed in manual modeling, minimization of errors due to human interference, and possibility of automating the testing process. Moreover, a standard document is used by many organizations around the world, and these benefits can be extended to all of them. Another positive trait of the approach is that the use of structural information of the text during information extraction process eliminates the need for rewriting sentences, as required by some available approaches.

Some limitations to the semi-automatic generation are the treatment of self-loops and of the hierarchical relationships, already identified in the literature. In the proposed approach, self-loops could be treated if they were explicitly described in the text.

As future work, we intend to generate models for other services described in the PUS document, and evaluate the prototype from the user perspective, aiming at identifying usability issues and other problems that might be fixed in the final version of the tool.

Acknowledgment

The authors would like to thank Ana Maria Ambrósio, from the National Institute for Space Research, for providing requirements for service implementation, Emília Villani, from the Aeronautics Technology Institute, Cecília M. F. Rubira, from the University of Campinas, Patrick Henrique Brito, from the Federal University of Alagoas, Marco Vieira, from Coimbra University and Raphael Winckler de Bettio, from the University of Lavras, for their valuable help throughout this research. The authors would also like to thank Fundação de Amparo à Pesquisa do Estado de Minas Gerais - Fapemig,- Fundação de Apoio ao Ensino, à Pesquisa e à Extensão - FAEPEX/Unicamp and Institute of Computing - Unicamp, for providing financial support.

References

[1] V. Ambriola and V. Gervasi, "Processing natural language requirements," in In Proceedings of ASE 1997. IEEE Press, 1997, pp. 36–45.

[2] K. Pohl, G. Böckle, and F. J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[3] R. Pressman, Engenharia de software. McGraw-Hill, 2006.

[4] T. Yue, L. Briand, and Y. Labiche, "Automatically deriving a UML analysis model from a use case model," Simula Research Laboratory, Tech. Rep. 2010-15 (Version 2), 2013.

[5] D. K. Deeptimahanti and R. Sanyal, "Semi-automatic generation of UML models from natural language requirements," in Proceedings of the ISEC '11. New York, NY, USA: ACM, 2011, pp. 165–174, retrieved: 2015.09.17. [Online]. Available: http://doi.acm.org/10.1145/1953355.1953378

[6] L. Kof, "Translation of textual specifications to automata by means of discourse context modeling," in Requirements Engineering: Foundation for Software Quality, ser. LNCS, M. Glinz and P. Heymans, Eds. Springer Berlin Heidelberg, 2009, vol. 5512, pp. 197–211.

[7] ECSS, Ground systems and operations - Telemetry and telecommand packet utilization. ECSS-E-70-41A. ESA Publications Division, 2003.

[8] V. S. Alagar and K. Periyasamy, Specification of Software Systems. Springer London Dordrecht Heidelberg New York, 2011.

[9] L. M. Northrop, "SEI's software product line tenets," IEEE Softw., vol. 19, no. 4, Jul. 2002, pp. 32–40, retrieved: 2015.09.17. [Online]. Available: http://dx.doi.org/10.1109/MS.2002.1020285

[10] J. van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on, 2001, pp. 45–54.

[11] K. Kang and H. Lee, "Variability modeling," in Systems and Software Variability Management, R. Capilla, J. Bosch, and K.-C. Kang, Eds. Springer Berlin Heidelberg, 2013, pp. 25–42.

[12] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11231, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Tech. Rep. CMU/SEI-90-TR-021, 1990, retrieved: 2015.09.17.

[13] K. Czarnecki, "Generative programming: Principles and techniques of software engineering based on automated configuration and fragment-based component models," Ph.D. dissertation, Technical University of Ilmenau, October 1998.

 ISBN: 978-1-61208-438-1

[14] K. Czarnecki, S. Helsen, and E. Ulrich, "Staged configuration through specialization and multilevel configuration of feature models," Software Process: Improvement and Practice, vol. 10, 04/2005 2005, pp. 143 – 169.

[15] E. Charniak, "Statistical techniques for natural language parsing," AI Magazine, vol. 18, no. 4, 1997, pp. 33–44.

[16] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in Meeting of the Association for Computational Linguistics, 2003, pp. 423–430.

[17] T. Yue, L. Briand, and Y. Labiche, "An automated approach to transform use cases into activity diagrams," in Modelling Foundations and Applications, ser. LNCS, T. Kühne, B. Selic, M.-P. Gervais, and F. Terrier, Eds. Springer Berlin Heidelberg, 2010, vol. 6138, pp. 337–353, retrieved: 2015.09.17. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13595-8_26

[18] ——, "A use case modeling approach to facilitate the transition towards analysis models: Concepts and empirical evaluation," in Model Driven Engineering Languages and Systems, ser. Lecture Notes in Computer Science, A. Schürr and B. Selic, Eds. Springer Berlin Heidelberg, 2009, vol. 5795, pp. 484–498.

[19] T. Yue and S. Ali, "Bridging the gap between requirements and aspect state machines to support non-functional testing: Industrial case studies," in Modelling Foundations and Applications, ser. Lecture Notes in Computer Science, A. Vallecillo, J.-P. Tolvanen, E. Kindler, H. Störrle, and D. Kolovos, Eds. Springer Berlin Heidelberg, 2012, vol. 7349, pp. 133–145, retrieved: 2015.09.17. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31491-9_12

[20] S. Ali, T. Yue, L. Briand, and S. Walawege, "A product line modeling and configuration methodology to support model-based testing: An industrial case study," in Model Driven Engineering Languages and Systems, ser. Lecture Notes in Computer Science, R. France, J. Kazmeier, R. Breu, and C. Atkinson, Eds. Springer Berlin Heidelberg, 2012, vol. 7590, pp. 726–742, retrieved: 2015.09.17. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33666-9_46

[21] D. K. Deeptimahanti and M. A. Babar, "An automated tool for generating UML models from natural language requirements," in Proceedings of ASE'09, Nov 2009, pp. 680–682.

[22] L. Kof and B. Penzenstadler, "Faster from requirements documents to system models: Interactive semi-automatic translation," in Proceedings of the REFSQ 2011 Workshops REEW, EPICAL and RePriCo, the REFSQ 2011 Empirical Track (Empirical Live Experiment and Empirical Research Fair), and the REFSQ 2011 Doctoral Symposium, B. Berenbach, M. Daneva, J. Dör, S. Fricker, V. Gervasi, M. Glinz, A. Herrmann, B. Krams, N. H. Madhavji, B. Paech, S. Schockert, and N. Seyff, Eds. ICB Research Report- 44, 2011, pp. 14–25.

[23] V. A. d. Santiago Júnior and N. Vijaykumar, "Generating model-based test cases from natural language requirements for space application software," Software Quality Journal, vol. 20, no. 1, 2012, pp. 77–143.

[24] V. A. d. Santiago Junior, N. Vijaykumar, D. Guimaraes, A. Amaral, and E. Ferreira, "An environment for automated test case generation from statechart-based and finite state machine-based behavioral models," in Proceedings of ICSTW '08, April 2008, pp. 63–72.

[25] R. Lutz, "Software engineering for space exploration," Computer, IEEE Computer Society, vol. 44, no. 10, 2011, pp. 41–46, retrieved: 2015.09.17.

[26] J. Penã, M. G. Hinchey, A. Ruiz-Cortés, and P. Trinidad, "Building the core architecture of a nasa multiagent system product line," in AOSE, ser. LNCS, L. Padgham and F. Zambonelli, Eds. Springer Berlin Heidelberg, 2007, vol. 4405, pp. 208–224.

[27] K. Weiss, "Reviewing aerospace proposals with respect to software architecture," in Aerospace Conference, 2007 IEEE, March 2007, pp. 1–20.

[28] J. Fant, H. Gomaa, and R. Pettit, "Software product line engineering of space flight software," in Product Line Approaches in Software Engineering (PLEASE), 2012 3rd International Workshop on, June 2012, pp. 41–44.

[29] I. Habli, T. Kelly, and I. Hopkins, "Challenges of establishing a software product line for an aerospace engine monitoring system," in Software Product Line Conference, 2007. SPLC 2007. 11th International, Sept 2007, pp. 193–202.

[30] J. G. Greghi, E. Martins, and A. M. B. R. Carvalho, "Semi-automatic generation of extended finite state machines from natural language standard documents," in Dependable Systems and Networks Workshops (DSN-W), 2015 IEEE International Conference on, June 2015, pp. 45–50.

[31] C. D. Manning and H. Schütze, Foundations of statistical natural language processing. MIT press, 1999.

[32] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in Proceedings of the NAACL '03 - Volume 1. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 173–180, retrieved: 2015.09.17. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073478

[33] R. P. Pontes, P. C. Véras, A. M. Ambrosio, and E. Villani, "Contributions of model checking and cofi methodology to the development of space embedded software," Empirical Software Engineering, vol. 19, no. 1, 2014, pp. 39–68, retrieved: 2015.09.18. [Online]. Available: http://dx.doi.org/10.1007/s10664-012-9215-y

[34] Research Group of Applied Software Engineering, "FAMA-FeAture Model Analyser," http://www.isa.us.es/fama/, retrieved: 2015.09.17.

 ISBN: 978-1-61208-438-1

An Approach to Compare UML Class Diagrams Based on Semantical Features of Their Elements

Oksana Nikiforova, Konstantins Gusarovs, Ludmila Kozacenko, Dace Ahilcenoka, Dainis Ungurs
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
{oksana.nikiforova, konstantins.gusarovs, ludmila.kozacenko, dace.ahilcenoka, dainis.ungurs}@rtu.lv

***Abstract* —Models are widely used in software engineering, where the Unified Modelling Language (UML) class diagrams are the top notation to present the core system structure and serves as the main artefact for analysis, design and implementation of the software system. As far as the UML class diagram is created at the different levels of abstraction, fluently modified and used to present different aspects of the system, the software development project may need to manage different versions of the system model presented in that notation. Therefore, it is very important to have an ability to compare different versions of the UML class diagram created for the same system to avoid duplicates, missings and contradictions in the whole system model. In this paper an approach to do such a comparison is being described and tested on a simple example in comparison with some other similar methods. We analyze some of the existing methods and algorithms used for the UML class diagram comparison and offer the new approach on a subject. The approach offered in this paper is based on the evaluation of semantical features of the UML class diagram elements.**

Keywords – semi-automatic diagram comparison; conformity verification; UML class diagram.

I. INTRODUCTION

Nowadays, system development starts with a modeling of a problem domain and then of a software domain. The benefit of using the models is that it helps to solve the complexity of systems by showing only required information and representing it in a graphical manner comprehensible to a human. Since modeling is used from the early software development phases, the system engineers can have a large amount of the model's versions representing the system from the different aspects, in different development stages and versions. In order to evaluate the differences between these model versions, one needs to compare them. These differences allow detecting the incomplete functionality, errors or lack of correspondence. For example, when it is necessary to find out if the model specified in documentation complies with the actual system model, which can be generated automatically from the code.

In addition, the comparison of the model versions can be used to analyze the differences between the implemented systems and systems under development, thus identifying the reusable components [1].

One more task where model comparison is of high importance is evaluation of model transformation itself. During the software development, the models can be created manually [2], generated from the code [3] or transformed from the other models, e.g., using the transformation approaches presented in [4]-[6]. The model comparison can be used to evaluate the models obtained automatically (generated from the code or via transformation) so that the model generation or transformation method can be validated [7][8]. In this case, a formal approach to the model comparison can serve to evaluate the method proposed and used for automatic generation of some diagram or model transformation. The manual model comparison is a time consuming and complicated task. Therefore, the automatic comparison is preferred.

Commonly, different graphical notations are used to describe the system or its part in different levels of abstraction. There are many notations that can be used to model system [9]. It can delay an evolution of the comparison methods used for the model conformity verification, because we would need many comparison methods specific to the certain modeling language. Still, it is possible to try to introduce the method for evaluation of the most popular modeling language. One of these notations is the UML, which is recognized as an industry standard proposed by the Object Management Group [10]. The UML is designed to model and visualize the system from the different point of views, such as the system structure and behavior. The most widely used UML diagram is the class diagram, therefore the main focus of this paper is turned to the UML class diagrams and their comparison abilities. The goal of this paper is to propose an approach for the comparison of the UML class diagrams adoptable also for the other modeling languages, which have the similar infrastructure as UML.

The rest of the paper is structured as follows. The second section describes related work on existing model comparison methods and techniques. The third section explains the comparison approach offered by the authors. The proposed approach is demonstrated on an abstract example in the fourth section, where the defined calculations are applied to compare two class diagrams containing all the possible features to show the essence of the approach. The conclusions are made in the fifth section.

II. RELATED WORK

In order to cover the state of the art in the existing methods for UML class diagram comparison, the authors conducted a research using online libraries, such as IEEE,

 ISBN: 978-1-61208-438-1

EBSCO and Springer Link. Several methods exist in the area and approaches proposed differ in the results obtained from an UML-model comparison process (e.g [1][11]-[13]). Analyzing those methods we have searched for the ones that are providing the numerical metrics that describe model differences in order to compare those to our proposed approach. As a result two similar methods were selected for evaluation.

The first method similar to proposed by the authors is described by Mojeeb Al-Rhman Al-Khiaty and Moataz Ahmed [1]. The method is based upon several similarity metrics described as follows:

- **Shallow Lexical Name Similarity Metric (NS)** – describes the difference between two semantically similar class names.
- **Attribute Similarity Metric (ASim)** – describes the difference between two sets of class attributes.
- **Operations' Similarity Metric (OSim)** – describes the difference between two sets of operations (methods).
- **Internal Similarity Metric (IS)** – utilizes two previously defined **ASim** and **OSim** metrics in order to estimate the difference between two classes.
- **Neighborhood Similarity Metric (NHS)** – describes the difference of class neighborhoods (i.e., related classes) using special relation type comparison table.

All metrics defined above are being used to produce a similarity score for pairs of elements in the compared class diagrams.

The second method described in this paper is proposed by D. H. Qiu, H. Li, and J. L. Sun [11]. The authors of this paper propose not to compare class names while estimating the difference between two class models since it may result in a rather big impact to the comparison results. Similarly to [1], this method uses attribute and operation sets to define difference between compared class structure, however, relation similarity estimation is different – focusing on three types of class relations defined by the authors:

- **Inheritance** – which includes both inheritance and realization.
- **Method coupling** – when class A uses methods of class B that is commonly referred as a dependency.
- **Data coupling** – when class A uses publicly available data of class B, as well as cases of aggregation and composition.

As a result, a single number describing two class diagram similarity is obtained.

III. PROPOSED COMPARISON METHOD

In order to successfully compare two different UML class diagrams, it is necessary to take into account its elements, relations between them, as well as semantical information of those. Since the UML class diagrams are usually produced by the human system analysts, it is possible that two elements that are equal by their semantics have different names, which makes the naive approach not applicable. The authors state that the UML class diagram comparison should also be done by a human (however, it is possible to introduce some kind of automation) after the semantically equal element pairs are identified.

The proposed method compares the following of the UML class diagram elements [3]:

- Classes (and interfaces).
- Class attributes.
- Methods.
- Relations between elements.

For each of those elements the following comparison algorithm is defined:

1. Pair the elements from two diagrams according to their semantical meaning. This step of the algorithm requires human involvement.
2. Calculate distance between the elements of each pair.
3. Add the calculated distance to a model difference vector that is used to estimate the final difference.

After these steps are done, a vector containing distances between appropriate element pairs is constructed, and its length is being estimated to receive the resulting difference.

The distances between the classes and interfaces are calculated using Table I.

TABLE I. CLASS AND INTERFACE DISTANCES

Criteria	Distance
In both models semantically equal elements with same names are present	0
In both models semantically equal elements are present, however, their names differ	0.5
One of the model doesn't contain semantically equal class from another model	1

In order to calculate the distances between the class attributes, it is necessary to construct the temporary vector shown in formula 1 and estimate its length (described in details in Table II).

$$\langle a|s|n|t\rangle \quad (1)$$

TABLE II. ELEMENTS OF COMPARISON VECTOR FOR CLASS ATTRIBUTES

Element	Criteria	Value
a	Difference between access modifiers of appropriate class attributes	0 for the same, 1 for different
s	Static modifier flag	0 if both attributes share the same static modifier, 1 otherwise
n	Name difference	0 for the same attribute names, 1 for different
t	Attribute type difference	0 for the same type, 1 for different

In case one of the attributes is present only in one of the compared class diagrams (or when the enclosing class is not present), all the elements of attribute difference vector are set

 ISBN: 978-1-61208-438-1

to 1. After the construction of the vector, its length is being calculated providing distance value between attributes.

The distance calculation between the class methods also requires the construction of temporary vector (formula 2) and its length estimation (described in details in Table III):

$$\langle o|a|s|n|p|r\rangle \quad (2)$$

TABLE III. ELEMENTS OF COMPARISON VECTOR FOR CLASS METHODS

Element	Criteria	Value
o	Owning class difference	If a method is defined in a semantically equal classes (interfaces) – 0, 1 otherwise
a	Difference between access modifiers	0 for the same, 1 for different
s	Static modifier flag	0 if both methods share the same static modifier, 1 otherwise
n	Name difference	0 for the same method names, 1 for different
p	Difference between method arguments	0.2 for each mismatching attribute type, 0.5 for missing argument (see explanation below)
r	Difference between return type	0 when return type is semantically equal, 1 otherwise.

The difference between the method arguments is calculated basing on the types of arguments. In order to calculate this difference, the arguments of the compared methods are paired by their semantical meaning, and then for the each pair the types of the arguments are being compared. If the types mismatch, 0.2 is being added to the difference. In case when the argument is present only in one of the compared methods, the difference is increased by 0.5 thus giving the formula 3.

$$p = 0.2 * a_t + 0.5 * a_m \quad (3)$$

Where:

a_t – number of the method arguments with mismatching types.

a_m – number of the cases when the method argument is present only in the one of compared UML class diagrams.

The argument order is not being taken into account, since the argument pairing by their semantical meaning is performed before the actual difference calculation.

The relation comparison is also done using the difference vector shown in formula 4 that is described as follows with the detailed explanation given in the Table IV.

$$\langle s|t|y|m\rangle \quad (4)$$

After the comparison of the identified element pairs, set of distances between those is received. This set of values is then converted into n-dimension model difference vector, where n is a number of the identified element pairs. The final model difference estimation is equal to the length of the model difference vector and is represented by a single number.

TABLE IV. RELATION COMPARISON VECTOR ELEMENTS

Element	Criteria	Value
s	Relation source difference – denotes if relation is outgoing from the semantically equal class in both models	0 for the same class, 1 for different
t	Relation target difference – denotes if relation is incoming into the semantically equal class in both models	0 for the same class, 1 for different
y	Relation type difference	0 if both relations are of the same type, 1 otherwise
m	Multiplicity difference	0 if relations have the same multiplicity, 1 otherwise

In all the cases above, when the n-dimensional vector length is mentioned, it is calculated by the following formula 5 (Euclidian distance).

$$l = \sqrt{\sum_{i=1}^{n} e_i^2} \quad (5)$$

Thus, the final output of the proposed UML class diagram comparison method is the number which defines the distance between the diagrams that are compared. The larger is the resulting number, the more differences are noted. Such information is useful when developing model transformations with the target of the UML class diagram or code – thus generated model/code can be compared to the ones produced by a human in order to define the quality of transformation. The shorter is the distance from the generated class diagram to the etalon, the higher is a quality of the defined transformation.

It is also possible to use the model difference vector in order to detect changes when working with several versions of the same UML class diagram. In such case each element of this vector determines the amount of changes for each of the UML class diagram elements that are being compared. It is also possible to apply different weights to the different elements of the model difference vector however there is no universal solution for the weighting in this case.

 ISBN: 978-1-61208-438-1

IV. APROBATION OF THE METHOD

In order to test the proposed UML class diagram comparison method, three simple UML class diagrams were created. The diagrams contain 2 classes: Point and Line, and describe the abstract geometrical domain. The class Line consists of two points – the start and the end. The first class diagram is shown in Figure 1 and is used as a reference diagram in the comparison. It means that two other diagrams are compared vice versa of this.

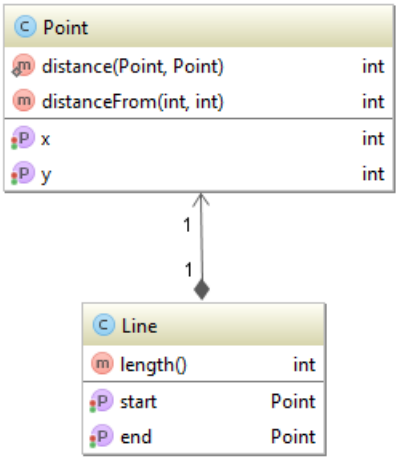


Figure 1. Reference UML class model (Diagram 1).

The second class diagram shown in Figure 2 is different from the first one in two aspects:
1) the class name –`Point` is renamed to `Coordinates`
2) the difference in arguments of the method `Coordinates.distanceFrom()`.

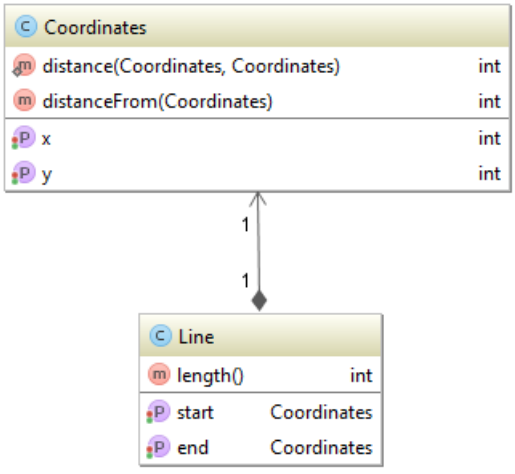


Figure 2. Class model with renamed class (Diagram 2).

The third diagram is shown in Figure 3, while it shares the same class names it has different return types for methods that are used to calculate distance between two points – the methods `Point.distance()` and the `Line.length()` respectively. Also, the arguments of the method `Point.distanceFrom()` are different in the same way as in the diagram in Figure 2.

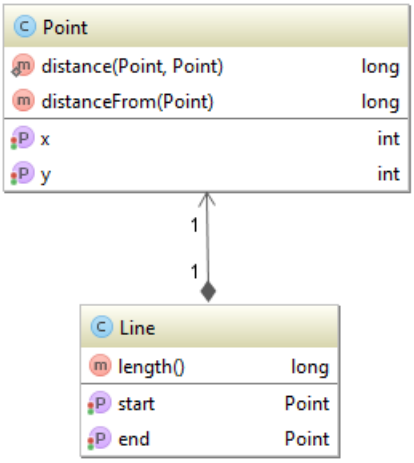


Figure 3. Class model with different return type (Diagram 3).

A. Comparison of Diagram 1 and Diagram 2

Comparison of the UML class diagrams using the approach offered in this paper requires the identification of the element pairs and calculation of the distance between them. In this paper the *accessor* and *mutator* methods are being omitted since the distance between them is equal to 0 due to equality of the names, access modifiers, return types and signatures. The details in comparison of the element pairs and the distance are shown in Table V.

The estimation of the model difference vector for those two models gives the final model difference equal to **1.5811** (formula 6).

$$Difference = \sqrt{\sum_{i=1}^{10} x_i^2} = \sqrt{0.5^2 + 1.5^2} = \sqrt{2.5} \approx 1.5811 \quad (6)$$

B. Comparison of Diagram 1 and Diagram 3

The elements of the diagram difference vector for the UML class diagrams 2 and 3 as well, as appropriate diagram element pairs (those that are responsible for these element values) are shown in Table VI.

The estimation of the diagram difference vector for those two models gives the final model difference equal to **2.0616** (formula 7).

$$Difference = \sqrt{\sum_{i=1}^{9} x_i^2} = \sqrt{1^2 + 1^2 + 1.5^2} = \sqrt{4.25} \approx 2.0616 \quad (7)$$

 ISBN: 978-1-61208-438-1

TABLE V. ELEMENT PAIR COMPARISON FOR DIAGRAM 1 AND DIAGRAM 2

Diagram 1 Element	Diagram 2 Element	Distance
`Point`	`Coordinates`	0.5
`Point.x`	`Coordinates.x`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Point.y`	`Coordinates.y`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Point.distance()`	`Coordinates.distance()`	$Len(\langle 0\|0\|0\|0\|0\|0\rangle) = 0$
`Point.distanceFrom()`	`Coordinates.distanceFrom()`	$Len(\langle 0\|0\|0\|1.5\|0\|0\rangle) = 1.5$
`Line`	`Line`	0
`Line.start`	`Line.start`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Line.end`	`Line.end`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Line.length()`	`Line.length()`	$Len(\langle 0\|0\|0\|0\|0\|0\rangle) = 0$
`Aggregation (Line -> Point)`	`Aggregation (Line -> Coordinates)`	$Len(\langle 0\|0\|0\|0\rangle) = 0$

TABLE VI. ELEMENT PAIR COMPARISON FOR MODELS 1 AND DIAGRAM 3

Diagram 1 Element	Diagram 3 Element	Distance
`Point`	`Point`	0
`Point.x`	`Point.x`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Point.y`	`Point.y`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Point.distance()`	`Point.distance()`	$Len(\langle 0\|0\|0\|0\|0\|1\rangle) = 1$
`Point.distanceFrom()`	`Point.distanceFrom()`	$Len(\langle 0\|0\|0\|1.5\|0\|0\rangle) = 1.5$
`Line`	`Line`	0
`Line.start`	`Line.start`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Line.end`	`Line.end`	$Len(\langle 0\|0\|0\|0\rangle) = 0$
`Line.length()`	`Line.length()`	$Len(\langle 0\|0\|0\|0\|0\|1\rangle) = 1$
`Aggregation (Line -> Point)`	`Aggregation (Line -> Point)`	$Len(\langle 0\|0\|0\|0\rangle) = 0$

C. *Result Analysis*

The analysis of the results achieved proves to be as expected: Diagram 1 and Diagram 2 are actually less different then Diagram 1 and Diagram 3 despite the fact that in Diagram 2 the class Point has the different name. This is due to the class Point/Coordinates itself is semantically the same in both Diagrams 1 and 2, i.e., with the same attributes and methods. Therefore, the impact on the class difference is much slighter.

Such results seem to be relevant in case of studying the output of the human-produced class diagrams that are commonly used in the first stages of a software development process. Since the human system analysts may (and usually will) use different names for the similar concepts when modeling the problem domain class, the name difference should affect comparison results in a slightly lower way than the structural difference of compared models.

In comparison to the proposed approach method described by Mojeeb Al-Rhman Al-Khiaty and Moataz Ahmed [1] tends to define more differences between models in example case – due to use of Longest Common Subsequence (LCS) algorithm when comparing the names of the model elements. Exact numbers aren't provided in the paper due to different scales of the numbers.

D. H. Qiu's, H. Li's, and J. L. Sun's method [11] was also compared to the proposed one. In this case name differences aren't taken into account thus method shows less differences between compared class models – only ones that are result of inner structure mismatch.

Thus we can conclude that proposed method is somewhere between those two eliminating the drawbacks of former.

V. CONCLUSION

One of the recent trends used in the iterative software development is a model presenting the system at the different levels of abstraction. As the system model is created at the different stages of the system development and in the different manner – manually or generating from some text information or other model, there is a need to evaluate the current version of some diagram and compare it to the other diagrams created at the previous stages of the project or in the different way of the modelling.

The most widely used notation in the modern software development projects is the UML, and its class diagram is applicable at the different abstraction levels of the software system development. Therefore, the most important task of the comparison of two models is exactly the UML class diagram comparison and evaluation. An effort to find a suitable approach to compare two UML class diagrams in advanced scientific databases gave the authors very pure results. Namely, there are a very few methods how to compare the UML class diagrams and they don't provide a valuable result.

 ISBN: 978-1-61208-438-1

The authors of this paper are working on the development of the model transformation method for the generation of the UML class diagram from the so-called two-hemisphere model [5]. There is a need to compare the received UML class diagram with the diagram created manually during the software development process to approve the quality of the transformation offered. This is one more reason to turn the attention to searching for existing approach to the UML diagram comparison or inventing a new one.

The comparison approach offered in this paper is based on the semantical features of the elements presented in the UML class diagram and takes into consideration the structural facilities of the diagram as they are more essential than, e.g., the name differences. The essence of the approach is based on the identification of the semantically same or similar pairs of the diagram elements and further evaluation of the distance between them.

The comparison approach offered in this paper is applied to the several examples to compare the class diagrams created in the different manner, but, due to the length limitations of the paper, only the abstract example is demonstrated here. The application of the comparison approach to the evaluation of the transformations defined by the two-hemisphere model-driven approach is stated as a direction for the future research.

ACKNOWLEDGMENT

The research presented in the paper is supported by the Latvian Council of Science, No. 342/2012 "Development of Models and Methods Based on Distributed Artificial Intelligence, Knowledge Management and Advanced Web Technologies".

REFERENCES

[1] Al-Khiaty, M.A.-R.; Ahmed, M., "Similarity assessment of UML class diagrams using simulated annealing," Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on , vol., no., pp.19,23, 27-29 June 2014

[2] Sharifi H.R., Mohsenzadeh M., Hashemi S.M. CIM to PIM Transformation: An Analytical Survey. International Journal of Computer Technology & Applications. 2012, vol.3, no.2, pp.791-796. ISSN: 2229-6093.

[3] Brambilla M., Cabot J., Wimmer M. Model-Driven Software Engineering in Practice. 1edition. USA: Morgan & Claypool Publishers, 2012.

[4] Al-Jamini H., Ahmed M. Transition from Analysis to Software Design: A Review and New Perspective. The Proceeding of International Conference on Soft Computing and Software Engineering. 2013, vol.3, no.3, pp. 169-176.

[5] Nikiforova, O., Gusarovs, K., Gorbiks, O., Pavlova N. BrainTool A Tool for Generation of the UML Class Diagrams. In: Proceedings of the Seventh International Conference on Software Engineering Advances : The Seventh International Conference on Software Engineering Advances (ICSEA 2012), Lisbon, Portugal, 18-23 Novemer, 2012. Lisbon: IARIA, 2012, 60-69.lpp.

[6] Rodriguez-Dominguez, C., Ruiz-Lopez, T., Benghazi, K., Noguera, M., u.c. A Model-Driven Approach for the Development of Middleware In: Technologies for Ubiquitous Systems. 9th International Conference on Intelligent Environments (IE), Athens, Greece, 16-17 July, 2013. IEEE, 2013, pp.16-23.

[7] Kriouile A., Gadi T., Balouki Y. IM to PIM Transformation: A criteria Based Evaluation. International Journal of Computer Technology & Applications. 2013, vol.4, no.4, pp.616-625.

[8] Lano K., Kolahdouz-Rahimi S., Poernomo I. Comparative Evaluation of Model Transformation Specification Approaches. International Journal of Software and Informatics. 2012, vol.6, no.2, pp. 233-269.

[9] Harmon, P, Wolf, C. The State of Business Process Management 2014 [online]. BPTrends, 2014 [viewed 19 April 2014]. Available from: http://www.bptrends.com/bpt/wp-content/uploads/BPTrends-State-of-BPM-Survey-Report.pdf

[10] Unified Modeling Language: superstructure v.2.2, OMG. Available: http://www.omg.org/spec/UML/2.2/Superstructure [retrieved: August, 2014].

[11] Qiu, D.H.; Li, H.; Sun, J.L., "Measuring software similarity based on structure and property of class diagram," Advanced Computational Intelligence (ICACI), 2013 Sixth International Conference on , vol., no., pp.75,80, 19-21 Oct. 2013

[12] Maoz, S.; Ringert, J.O.; Rumpe, B., "CDDiff: Semantic Differencing for Class Diagrams", ECOOP 2011 – Object-Oriented Programming, 25th European Conference, Lancaster, Uk, pp.230-254, 25-29 July, 2011.

[13] Uhrig S., "Matching class diagrams: with estimated costs towards the exact solution?.", 2008 international workshop on Comparison and versioning of software models (CVSM '08), pp. 7-12, ACM, New York, NY, USA, 2008.

 ISBN: 978-1-61208-438-1

Model-Based Evaluation and Simulation of Software Architecture Evolution

Peter Alexander

Thai-German Graduate School of Engineering
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
email: peter.a-sse2013@tggs-bangkok.org

Ana Nicolaescu, Horst Lichter

RWTH Aachen University
Research Group Software Construction
Aachen, Germany
email: {nicolaescu, lichter}@swc.rwth-aachen.de

***Abstract*—The software architecture description is often the reasoning basis for important design decisions. Nevertheless, during the evolution of a system, the software architecture tends to deviate from its description which gradually approaches obsolescence. Software architecture reconstruction tools can be employed to retrieve up-to-date descriptions, however reconstruction by itself is never a purpose. The reconstructed architecture description should, e.g., support the architects to identify the best evolution variant with respect to a set of quality characteristics of interest. The state of the art approaches address reconstruction and evolution simulation in separation. To simulate changes, the current state of the system must be first manually modeled. In our previous work, we presented ARAMIS, an approach to support the reconstruction and evaluation of software architecture with a strong emphasis on software behavior. In this paper, we propose the extension of our approach for enabling the simulation of design decisions on the recovered architecture description. To reduce complexity and support a more focused analysis, we allow to specify and apply viewpoints, views, and perspectives on the recovered description and its evolution simulations.**

Keywords–Software Architecture Reconstruction; Software Architecture Evaluation; Software Architecture Simulation; Software Architecture Viewpoint; Software Architecture

I. INTRODUCTION

As abstractions of the architecture of a software system, the prescriptive (as-designed) and descriptive (as-implemented) architecture descriptions can greatly contribute to support reasoning and evolution. However, very often design decisions are not documented in the descriptive architecture description [1], which consequently degrades [2] and gradually approaches obsolescence. In consequence, further design decisions may be taken based on low-fidelity reasoning. This situation can easily lead to the continuous degradation of the system's quality as formulated in Lehman's laws of software evolution [3].

To avoid this, the evolution should be analyzed using up-to-date descriptive architecture descriptions. However, the software architecture encompasses a wide variety of aspects, making it very difficult to explore it in its entirety. To enable a more focused analysis, the concepts of view and viewpoint have been introduced and adopted by the major architecture description standards (i.e., [4], [5]). Thus, an architecture viewpoint (e.g., operational, deployment, logical) represents "a set of conventions for constructing, interpreting, using, and analyzing one type of Architecture View". An architecture view "expresses the Architecture of a System of Interest" from the perspective of several stakeholders using the conventions of its corresponding viewpoint. According to [6], a perspective is system-independent and specifies a further refinement of a view according to a set of interesting quality properties.

Several view models have been proposed (e.g., [7], [8]) to guide the description of software architectures. However, most of them make a clear distinction between static and dynamic views. The dynamic view is usually very complex, bloated with huge low-level run-time information, making it hard to document (and thus is often avoided) and understand. We strongly argue that the dynamic view should be considered as a description by itself and be considered from various viewpoints and perspectives.

Given that the behaviour of a system is the one that actually supports its use-cases, we think that evolution should also be discussed at this level. The viewpoints and perspectives can be then applied to support the analysis with relevant information while avoiding cluttering.

In our previous work we proposed the Architecture Analysis and Monitoring Infrastructure (ARAMIS), an approach for the reconstruction and evaluation of software architectures with a strong emphasis on software behavior. Our approach allows to enrich the architecture description with the software's static and dynamic information and identify architectural degradation. In our previous work [9], we presented our achievement in developing a semi-automatic approach to unobtrusively extract the run-time interactions of a software system, map these on architecture-level, identify unintended architectural behavior and mark these as violations, and characterize the system's behavior using a series of architectural metrics. This paper presents an extension of our approach to allow the simulation of design decisions and the viewpoint-based analysis.

The remainder of this paper is organized as follows: in Section II, we describe the ARAMIS meta-model and our proposed viewpoint-supported evolution process; Section III offers an overview of related work; Section IV discusses future work and concludes the paper.

II. CONCEPT

ARAMIS aims to support the systematic evolution of software architecture through the process depicted in Figure 1. We elaborate the process in four sequential sessions (monitoring, analysis, evaluation, and evolution) in which a cycle of activities (A) are divided between the architect (upper row) and ARAMIS (lower row).

Monitoring. To allow ARAMIS to reconstruct the behavior of a software system, the architect needs to provide the system's run-time information obtained using run-time

 ISBN: 978-1-61208-438-1

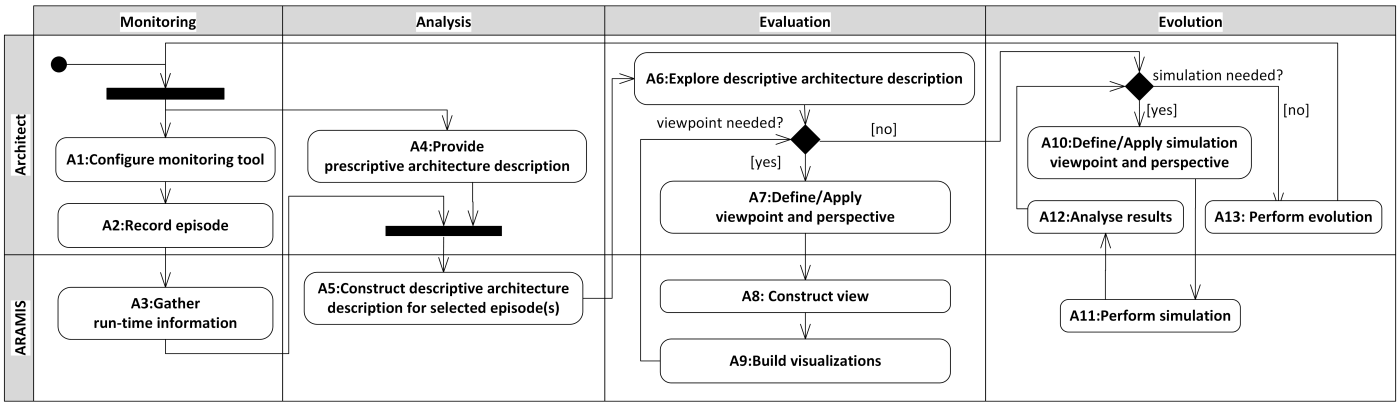


Figure 1. ARAMIS Process

monitoring. To avoid collecting large chunks of unnecessary information, the scope and granularity of the monitoring can be configured (A1). Various system episodes can then be executed, e.g., test cases and GUI interactions (A2). ARAMIS then intercepts the run-time interactions between the system's code building blocks and persists these in a single collection, a so-called episode, for later analysis (A3).

Analysis. By providing the system's prescriptive architecture description (A4) and selecting the episodes to be analyzed, the architect can trigger ARAMIS to reconstruct the system's descriptive architecture description (A5). Based on this, ARAMIS then maps the code-level interactions from the chosen episodes on architecture-level units from the prescriptive architecture and validates these against architectural rules. The result is then characterized using a series of architectural metrics and presented for evaluation.

Evaluation. Upon analyzing the presented result (A6), the architect might want to refine his view by applying a certain viewpoint and/or perspective on it (A7), as later exemplified in Subsection B. ARAMIS then refines the result accordingly (A8) and builds various relevant visualizations thereof (A9) in order to support reasoning of evolution variants.

Evolution. To assure that the evolved system will exhibit the quality properties of interest, ARAMIS allows the architect to simulate the considered design decisions by applying a mock-up architecture description on the current software behavior. The simulated result can then be further refined using viewpoints and perspectives (A10, A11). Iterative refinements of the design decisions (A12, A10, A11) can be performed through further simulations to gain the best evolution variant for evolving the system (A13).

All in all, this cyclic process embodies the continuous reconstruction, evaluation and evolution of software architecture.

A. ARAMIS - Meta-Model

In our previous work [9], we presented the meta-model of ARAMIS that can be used to create the system's prescriptive architecture description needed in A4. As depicted in Figure 2, we extended the meta-model to enable the definition of viewpoints and perspectives (A7, A10), but also abbreviated some parts which are loosely related to the focus of this paper.

The recorded episode from A2 produces a sequence of run-time traces, which are basically an ordered lists of so-called *Execution Record Pairs*. An execution record pair represents a one-way communication between a pair of *Code Building Blocks* which can be Java methods, classes, packages, etc, depending on the monitoring granularity configured in A1. This information serves as the system's low-level dynamic behavior and can be further processed by ARAMIS with the provision of the *Prescriptive Architecture Description* (A4).

The *Architecture Description* captures a set of *Architecture Units* (AU) and the *Communication Rules* between them. Each architecture unit can contain other architecture units and *Code Units* (CU), building a tree-like hierarchical structure. The code unit is a representative of a single code building block which participated during the run-time monitoring. It is designed to be untyped in order to make it programming-language independent and thus allowing further extension of ARAMIS to monitor other systems than Java-based system. Furthermore, the communication rule basically specifies pairs of AUs and the communication permission types between them (allowed/disallowed) (for a more detailed description, see [9]).

In order to focus the analysis on a specific type of system behavior, a *Viewpoint* can be defined. It is composed by one or more *Communication Patterns* which characterize the behavior of interest. The communication pattern between a pair of AUs specifies the communication direction and the number of communication hops - which is the number of intermediate AUs through which communication must pass between source and destination (e.g., all communications originating from some AU "X" and ending in some AU "Y" with 0 hops in between, i.e., only direct communication). It is worth noting that we designed the viewpoint to be scalable, allowing the architect to break down the analysis to any level of software architecture granularity. As a result, the viewpoint can support the system-independent analysis.

Furthermore, a *View* is obtained by applying a viewpoint on a concrete set of *Episodes* of a chosen *Software System*. To further refine the obtained result from various quality perspectives, *Perspectives* can be applied. We classify the perspective in four focuses: *Unit Involvement*, *Unit Interdependence*, *Communication Integrity*, and *Cardinality*. The unit involvement perspective focuses on identifying AUs that are considered as active (more outgoing than incoming communication) or passive (more incoming than outgoing communication). The interdependence type perspective focuses on identifying AUs depending on their coupling and cohesion

 ISBN: 978-1-61208-438-1

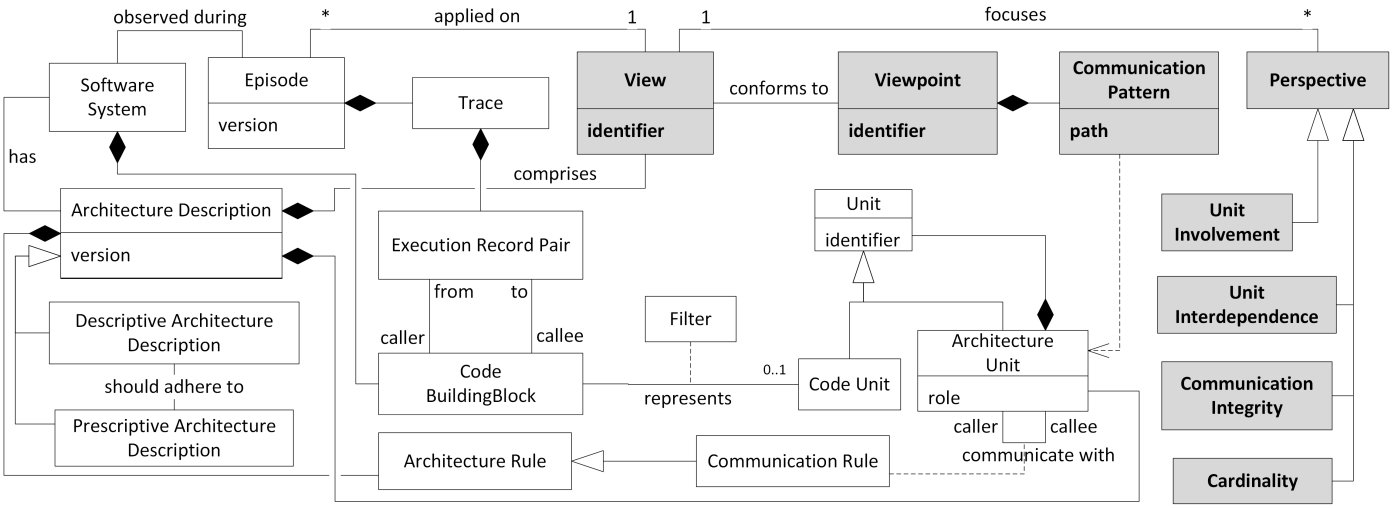


Figure 2. ARAMIS Meta-Model. The new elements of the meta-model are shaded.

attributes (e.g, highly coupled, low cohesion). The communication integrity perspective focuses on identifying violations of the pre-specified rules. Last but not least, the cardinality perspective focuses on quantifying the occurrences of a certain element - which can be AU, CU, or communication pattern - during the execution of some episodes to assess quality.

By versioning the architecture description and the episodes, ARAMIS allows the architect to apply the same viewpoint and perspective to obtain views that depict a system's past evolution or create simulations for the future. Furthermore, to support the analysis of evolution impact, ARAMIS enables the comparison of multiple views and highlight the differences between them (e.g., evolution of number of violations).

B. ARAMIS - Domain Specific Language

To support the modeling of viewpoints and perspectives (A7, A10) and the creation of views (A8, A11), we have designed a domain-specific language (DSL). We chose DSL as the modeling technique in order to provide high expressiveness in specifying the system behavior and high readability for the domain-experts. Also, with the ARAMIS-DSL, we intend to offer flexibility in extracting only architectural data interesting for a given purpose, thus reducing analysis complexity, especially in the case of large-scale software systems. Since this paper presents our work in progress, we do not explain the full-grammar of the DSL nor other implementation details. Instead, we exemplify the application of the viewpoint and view specification using an example case.

Example Case

Let "LayeredArchitectureSystem" be a layered system that the architect currently wants to analyze. It defines three layers: application layer (top), business layer (middle), and data layer (bottom). According to the layered design principles, each layer should depend on the layer immediately below it and the lower layers should not depend on any of the upper layers. The architect is worried that a cyclic dependency may have emerged between the business and data layers. An uncontrolled evolution in this direction would render the two layers monolithic.

Given his concern, the architect creates a viewpoint called "CyclicDependency" using the ARAMIS-DSL, as presented in Figure 3. The DSL keywords are made bold.

```
CyclicDependency := viewpoint {
   L1 := architecture unit with role 'Business Layer'
   L2 := architecture unit with role 'Data Layer'
   include CyclicCommunication := L1->L2->L1
}
```

Figure 3. ARAMIS Viewpoint Specification DSL Example

The viewpoint specifies two variables: *L1* and *L2*, which represent the "Business Layer" and "Data Layer" architecture units, respectively. Since the architect is interested in retrieving any cyclic communication between the two, he can *include* a communication pattern (named as "CyclicCommunication"), which starts from the *L1* (business layer) to the *L2* (data layer) and then forwarded back to the *L1* (business layer). The architect can then apply this viewpoint on any layered system to construct a view which specifically supports the cyclic dependency analysis between the business and data layers.

Furthermore, the ARAMIS-DSL supports the specification of views, which permits the architect to specify the system and corresponding episodes to be analyzed, to apply viewpoints and perspectives, and to perform evolution impact analysis. Our decision to place the perspective specification along with the view specification is based on our in line idea with the paradigm of architectural perspective introduced in [6], that the perspective aims to enhance the existing views to ensure that the architecture exhibits the desired quality properties and are therefore considered as 'orthogonal' to viewpoints.

In Figure 4, we exemplify the application of ARAMIS-DSL to specify two views: "CurrentBehavior" and "SimulatedBehavior". The "CurrentBehavior" view presents the actual system behavior, whereas the "SimulatedBehavior" presents the simulated system behavior. To these views, we apply the viewpoint "CyclicDependency" and perspective of cardinality to project the evolution of cyclic dependency occurrence.

For the sake of exemplifying, we assume that the architect previously monitored an episode called "aBusinessProcess".

 ISBN: 978-1-61208-438-1

```
analyze system LayeredArchitectureSystem
using viewpoint CyclicDependency
construct view CurrentBehavior       // actual view
with
   code unit version 1
   architecture unit version 1
   rule version 1
on episode aBusinessProcess version 1
construct view SimulatedBehavior     // simulation view
with
   code unit version 2
   architecture unit version 2
   rule version 1
on episode aBusinessProcess version 1
consider cardinality of CyclicCommunication
```

Figure 4. ARAMIS View and Perspective Specification DSL Example

The current prescriptive architecture description of the system is given by the triple (”code unit version 1”, ”architecture unit version 1”, ”rule version 1”). We assume that the architect has applied the viewpoint ”CyclicDependency” on this particular episode and architecture description, and consequently discovered many cyclic communications between the ”Business Layer” and ”Data Layer” architecture units. Therefore, the architect simulated merging some Java packages (thus creating ”code unit version 2”) and moving the newly created code unit to another layer (thus creating ”architecture unit version 2”), while the set of rules between the architecture units remained the same. The architect now wants to check how would the number of cyclic communication change if evolving the architecture as described in his simulation. He achieves this by applying the ”cardinality” perspective of ”CyclicCommunication” described in the used viewpoint, which quantifies the number of cyclic communications in each of the constructed views (”CurrentBehavior” and ”SimulatedBehavior”).

```
Project     : LayeredArchitectureSystem
Viewpoint   : CyclicDependency
Perspective : Cardinality
Result      :
> CurrentBehavior    : 100 CyclicCommunication
> SimulatedBehavior  :  30 CyclicCommunication
```

Figure 5. ARAMIS Simulation Result of the Example Case

The result, as it will be projected by ARAMIS is depicted in Figure 5. The ARAMIS simulation gives the architect that the design decisions he simulated in the ”SimulatedBehavior” will reduce the number of cyclic communications. The architect can further refine and re-simulate his design decisions to achieve a better result before bringing them into effect.

III. Related Work

The reconstruction of software behavior and up-to-date architecture description have been for long in the focus of software architecture community. However, little emphasis has been put on analyzing and validating the software dynamic behavior on various abstraction levels, which are defined in the static view of the architecture. For a more detailed overview of the existing works in this regard, see [9].

Proposals regarding the simulation of software architecture were also made. The simulation of various architectural design decisions by replicating the system’s behavior from UML diagrams have been offered (e.g., [10]–[12]). However, a study about continuous architecture simulation [13] has concluded that the modeling process of UML diagrams for evaluation purposes is very time-consuming and it makes the continuous simulation effort not appropriate for evaluating a software architecture. Other approaches (e.g., [14], [15]) allow to simulate the software architecture based on the software’s run-time behavior and mainly focus on some quality attributes like performance and availability. None of these simulation approaches focuses on the preliminary reconstruction of the architecture description as a basis for further analysis.

IV. Conclusion and Future Work

All in all, this paper presented and exemplified our current work to enable the viewpoint-based analysis and evolution of software architecture within the ARAMIS project. Our next steps are to conclude the implementation of the presented concept and to evaluate the achieved result.

References

[1] J. D. Herbsleb and D. Moitra, “Global software development,” IEEE Software, vol. 18, no. 2, Mar/Apr 2001, pp. 16–20.

[2] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, Software Architecture: Foundations, Theory, and Practice. Wiley Publishing, 2009.

[3] M. M. Lehman, “Laws of software evolution revisited,” in Proceedings of the 5th European Workshop on Software Process Technology. London: Springer-Verlag, 1996, pp. 108–124.

[4] M. W. Maier, D. E. Emery, and R. Hilliard, “Software architecture: Introducing IEEE standard 1471,” IEEE Computer, 2001.

[5] ISO/IEC/IEEE, “Systems and software engineering – architecture description,” ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 2011.

[6] N. Rozanski and E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2nd ed. Addison-Wesley, 2011.

[7] P. Kruchten, “The 4+1 view model of architecture,” IEEE Software, vol. 12, no. 6, Nov. 1995, pp. 42–50.

[8] C. Hofmeister, R. Nord, and D. Soni, Applied Software Architecture. Boston: Addison-Wesley, 2000.

[9] A. Dragomir, H. Lichter, J. Dohmen, and H. Chen, “Run-time monitoring-based evaluation and communication integrity validation of software architectures,” in the 21st Asia-Pacific Software Engineering Conference (APSEC 2014), Jeju, South Korea, December 1–4, 2014, vol. 1. IEEE, December 2014, pp. 191–198.

[10] A. Kirshin, D. Dotan, and A. Hartman, “A uml simulator based on a generic model execution engine,” in Proceedings of the 2006 International Conference on Models in Software Engineering. Springer-Verlag, 2006.

[11] V. Cortellessa, P. Pierini, R. Spalazzese, and A. Vianale, “Moses: Modeling software and platform architecture in uml 2 for simulation-based performance analysis,” in Proceedings of the 4th International Conference on Quality of Software-Architectures: Models and Architectures. Springer-Verlag, 2008.

[12] R. Singh and H. S. Sarjoughian, “Software architecture for object-oriented simulation modeling and simulation environments: Case study and approach,” Computer Science Engineering Dept., Arizona State University, Tempe, AZ, Tech. Rep., 2003.

[13] F. Mårtensson and P. Jönsson, “Software architecture simulation - a continuous simulation approach,” Master’s thesis, Blekinge Institute of Technology, 2002.

[14] V. Bogado, S. Gonnet, and H. Leone, “Modeling and simulation of software architecture in discrete event system specification for quality evaluation,” Simulation, vol. 90, no. 3, Mar. 2014, pp. 290–319.

[15] R. von Massow, A. van Hoorn, and W. Hasselbring, “Performance simulation of runtime reconfigurable component-based software architectures,” in Software Architecture - 5th European Conference, ECSA 2011, Essen, Germany, September 13–16, 2011. Proceedings. Springer-Verlag, 2011, pp. 43–58.

 ISBN: 978-1-61208-438-1

Towards Time-triggered Component-based System Models

Hela Guesmi, Belgacem Ben Hedia
CEA, LIST, PC 172
91191 Gif-sur-Yvette, France
Email: firstname.lastname@cea.fr

Simon Bliudze
EPFL IC IIF RiSD, Station 14
CH-1015 Lausanne, Switzerland
Email: simon.bliudze@epfl.ch

Saddek Bensalem, Jacques Combaz
Verimag,
38610 Gieres, France
Email: firstname.lastname@imag.fr

***Abstract*—In this paper, we propose a methodology for producing correct-by-construction Time-Triggered (TT) physical model by starting from a high-level model of the application software in Behaviour, Interaction, Priority (BIP). BIP is a component-based framework with formal semantics that rely on multi-party interactions for synchronizing components. Commonly in TT implementations, processes interact with each other through a communication medium. Our methodology transforms, depending on a user-defined task mapping, high-level BIP models where communication between components is strongly synchronized, into TT physical model that integrates a communication medium. Thus, only inter-task communications and components participating in such interactions are concerned by the transformation process. The transformation consists of: (1) breaking atomicity of actions in components by replacing strong synchronizations with asynchronous send/receive interactions, (2) inserting communication media that coordinate execution of inter-task interactions according to a user-defined task mapping, (3) extending the model with an algorithm for handling conflicts between different communication media and (4) instantiating task components and adding local priority rules for handling conflicts between inter-task and intra-task interactions. We also prove the correctness of our transformation, which preserves safety properties.**

Keywords–Anything; Time-Triggered paradigm; correct-by-construction; component-based design; model transformation; BIP Framework.

I. INTRODUCTION

A Time-Triggered (TT) system initiates all system activities -task activation, message transmission, and message detection- at predetermined points in time. Ideally, in a time-triggered operating system there is only one interrupt signal: the ticks generated by the local periodic clock. These statically defined activation instants enforce regularity and make TT systems more predictable than Event-Triggered (ET) systems. This approach is well-suited for hard real-time systems.

In [1] and [2], Kopetz presents an approach for real-time system design based on the TT paradigm which comprises three essential elements:

The global notion of time: It must be established by a periodic clock synchronization in order to enable a TT communication and computation,

The temporal control structure of each task: In a sequence of computational or communication processes (called *tasks*), the start of a task is triggered by the progression of the global time, independently from the involved data of the task. The worst-case execution time and thus the worst-case termination instant are also assumed to be known a priori. These statically predefined start and worst-case termination instants, define the temporal control structure of the task,

TT communication system: To isolate subsystems from each other in a TT architecture, a special interface called the temporal firewall has been designed. It consists in a shared memory element for unidirectional exchange of information between sender/receiver tasks components. It's the responsibility of the TT communication system [3], [1] to transport, with access to the global time, the information from the sender firewall to the receiver firewall. The instants at which information is delivered or received are a priori defined and known to all nodes. Furthermore, the TT communication service/protocol avoids interference between concurrent read and write operations on the memory elements.

Analysis and design of hard real-time systems often starts with developing a high-level model of the system. Building models allows designers to abstract away implementation details and validate the model regarding a set of intended requirements through different techniques such as formal verification, simulation, and testing. However, deriving a correct TT implementation from a high-level model is always challenging, since adding TT implementation details involves many subtleties that can potentially introduce errors to the resulting system. Furthermore, in the context of hard real-time systems (and time-triggered paradigm), services offered by target operating systems should be taken into account in the derived implementation.
Thus it is highly advantageous if designers can somehow derive a model with implementation details in a systematic and correct way from high-level models. We call such a model physical model. It can be automatically translated to the programming language specific to the target TT platform.

In this paper, we present a method for transforming high-level models in BIP [4] into TT physical model that integrates the three TT-paradigm properties mentioned above. The BIP framework is used for constructing systems by superposition of three layers: Behaviour, Interaction, and Priority. The Behaviour layer consists of a set of components represented by automata or Petri nets extended by data and functions given in C++. The interaction layer describes possible interactions between atomic components. Interactions are sets of ports allowing synchronizations between components. They are defined and graphically represented by connectors. The execution of interactions may involve transfer of data between the participating components. The third layer includes priorities between interactions using mechanisms for conflict resolution. In this work, we consider Real-Time BIP (RT-BIP) framework [5], [6] where behaviour of components is represented by a timed automaton [7].

We believe that the first two properties of the TT paradigm

 ISBN: 978-1-61208-438-1

can be obtained straightforwardly from RT-BIP, and that the main difficulty lies in representing tasks and inter-task communication system in the physical model. To achieve this goal, we introduced in this paper TT-BIP model, which combines elements of both RT-BIP, Send/Receive BIP [8] (SR-BIP) model and TT paradigm. It introduces additional structure in order to model the TT communication system explicitly. The TT communication system is modelled by introducing dedicated atomic components. Depending on a user-defined task mapping, components in the system executing the same task are grouped into composite components called *tasks*. The latter can interact only through the atomic components modelling the TT communication system. The TT-BIP model draws on some of the ideas behind SR-BIP – the model used for distributed implementation of untimed BIP systems [9] and RT-BIP systems [10]. We also define transformation rules for deriving a TT-BIP model from a high-level RT-BIP model and a task mapping. We show that this transformation preserves trace inclusion (i.e., safety properties).

The rest of this paper is structured as follows. In Section II, we present the basic concepts related to our work; RT-BIP and SR-BIP models. Section III formalizes the TT-BIP architecture and describes the transformation. Experimental results are presented in Section IV. Section V analyses related work. And finally, we make concluding remarks in Section VI. For lack of space, all correctness proofs appear in the appendix.

II. Background concept: BIP Framework

In this section, we first provide the definition and the semantics of RT-BIP. RT-BIP is a component framework for constructing systems by superposing three layers of modelling: Behaviour, Interaction, and priority. And then, we describe the SR-BIP model. Before giving the definition and semantics of an RT-BIP component, we first fix some notations.

A. Preliminary notations

1) Valuation function: Given a variable x, the domain of x is the set $D(x)$ of all values possibly taken by x. Given a set of variables X, We denote by $v(x)$ the corresponding element of $x \in D(x)$. A valuation of X is a function $v : X \to \bigcup_{x \in X} D(x)$ associating with each variable x its value $v(x)$. Given a subset of variables $X' \subseteq X$ and a variable value $a \in D(x)$, we denote by $v\,[X' \leftarrow a]$ the valuation defined by:

$$v[X' \leftarrow a](x) = \begin{cases} a & \text{if } x \in X' \\ v(x) & otherwise. \end{cases} \quad (1)$$

We denote by $\mathcal{V}(X) = \prod_{x \in X} D(X)$ the set of all possible valuations of the variables in X.

2) Guards: Guards are boolean expressions used to specify when actions of a system are enabled. A guard g is a predicate on a set of variables X. Given valuation $v \in \mathcal{V}(X)$, we denote by $g(v) \in \{\mathit{False}, \mathit{True}\}$ the evaluation of g for v.

3) Clocks: We assume that time progress is measured by clocks which are integer or real-valued variables increasing synchronously. Each clock can be reset (i.e., set to 0) independently of other clocks.
We denote by $\mathbb{R}_+$ the set of non-negative reals, and by $\mathbb{N}$ the set of non-negative integers. Given a set of clocks C, let $\mathcal{V}(C)$ be the set of all clock valuation functions $v_c : C \to \mathbb{R}_+$.
Given δ, such that $\delta \in \mathbb{R}_+$, for all $c \in C$, we use $c+\delta$ as usual notation for the valuation defined by $(v_c + \delta)(c) = v_c(c) + \delta$.

4) Timing constraints: Timing constraints are guards over the set of clocks C. They are used to specify when actions of a system are enabled regarding system clocks. The basic building blocks for timing constraints are comparisons; given a set of clocks C, $c \in C$ and $a \in \mathbb{R}_+$, comparison between the valuation of c and a can be presented as $c \sim a$ where $\sim \in \{\leq, <, =, >, \geq\}$. Constraints are built using the following grammar:

$$tc := \mathit{True} \mid \mathit{False} \mid c \sim a \mid tc \wedge tc \mid tc \vee tc \mid \neg tc$$

Notice that any guard tc can be written as:

$$tc := \bigwedge_{c \in C} l_c \leq c \leq u_c, \text{ where } l_c, u_c \in \mathbb{R}_+ \forall c \in C \quad (2)$$

We denote by $\mathcal{TC}(C)$, the set of clock constraints defined over clocks of C.

5) Time progress conditions: Time progress conditions are used to specify whether time can progress at a given state of the system. They correspond to a special case of timing constraint where $\sim$ is restricted to $\{\leq\}$ and operators $\neg$ and $\vee$ are disallowed. Formally, time progress conditions are defined by the following grammar:

$$tpc := \mathit{True} \mid \mathit{False} \mid c \sim a \mid tc \wedge tc, \text{ where } c \in C \text{ and } a \in \mathbb{R}_+$$

Note that any time progress condition tpc can be written as:

$$tpc = \bigwedge_{c \in C} c \leq u_c, \text{ where } u_c \in \mathbb{R}_+ \cup \{+\infty\} \quad (3)$$

B. Basic semantic model of RT-BIP

In RT-BIP, systems are built by composing *atomic components* with *interactions* defined using *connectors*.

A *component* in RT-BIP is essentially a timed automaton [11] labelled by ports that represent the component's interface for communication with other components. Let $\mathcal{P}$ be a set of ports. We assume that every port $p \in \mathcal{P}$ has an associated data variable x_p. This variable is used to exchange data with other components, when interactions take place.

***Definition** 1:* (Component). A component B is a tuple $B = (L, P, X, C, T, \{tpc_l\}_{l \in L})$ where: L is a finite set of control locations, $P \subseteq \mathcal{P}$ is a finite set of ports, called the interface of B, X is a set of local variables. We denote the set of variables associated to ports by $X_P \subseteq X$ and the set of the rest of local variables by X_B, such that $X = X_B \cup X_P$, C is a finite set of clocks, T is a set of labelled transitions. A transition $\tau \in T$ from a control location l to l' is a tuple $\tau = (l, p, g_\tau, f_\tau, R, l')$ where :

- p is a port.
- $g_\tau = g^X \wedge tc$ is a boolean guard, which is a conjunction of a predicate g^X on local variables X and a timing constraint tc over C. We say that a transition τ is enabled for the valuation $v \in \mathcal{V}(X)$, when its guard g_τ evaluates to *True*.
- f_τ is a function that updates the set of variables on X
- R is the subset of clocks $R \subseteq C$, that are reset by the transition τ.

For each place $l \in L$, tpc_l is a time progress condition.

***Definition** 2:* (Semantics of a component). The semantics of a component $B = (L, P, X, C, T, \{tpc_l\}_{l \in L})$ is defined as a labelled transition system $S_B = (Q_B, P_B, \xrightarrow[B]{})$, where: $Q_B =$

$L \times \mathcal{V}(C) \times \mathcal{V}(X)$ is the set of states, $P_B = P \cup \mathbb{R}_{\geqslant 0}$ is the set of labels: ports or time delays, $\xrightarrow[B]{} \subset Q_B \times P_B \times Q_B$ is the set of labelled transitions defined as follows. Let (l, v_c, v_x) and (l', v'_c, v'_x) be two states, $p \in P$, and $\delta \in \mathbb{R}_{\geqslant 0}$ be a delay.

- **Jump transitions**: We have $(l, v_c, v_x) \xrightarrow[B]{p} (l', v'_c, v'_x)$ iff there exists a transition $\tau = (l, p, g_\tau, f_\tau, R, l')$ enabled for (v_c, v_x) and $v'_x = f_\tau(v_x)$, and for all $c \in r$, $v'_c(c) = 0$.
- **Delay transitions**: We have $(l, v_c, v_x) \xrightarrow[B]{\delta} (l, v_c + \delta, v_x)$ iff $\forall \delta' \in [0, \delta]$, $tpc_l(v_c + \delta')$ evaluates to *True*.

A component B can execute a transition $\tau = (l, p, g_\tau, f_\tau, R, l')$ from a state (l, v_c, v_x) if its timing constraint is met by the valuation v_c. The execution of τ corresponds to moving from control location l to l', updating variables and resetting clocks of R. From state (l, v_c, v_x), B can also wait for $\delta > 0$ time units if the time progress condition tpc_l stays *True*. Waiting for δ time units increases all the clock values by δ. Notice that the execution of transitions is instantaneous and time elapses only on states.

The interaction model is specified by a set of interactions $\gamma \subseteq 2^P$. Interactions of γ can be enabled or disabled.

***Definition** 3:* (Interaction). Let $\{B_i\}_{i=1}^n$ be a set of components as above. An interaction α between components $\{B_i\}_{i=1}^n$ is a quadruple (a, X_a, G_a, F_a), where: $a \subseteq P$ contains at most one port of every component, that is, $\mid a \cap P_i \mid \leqslant 1$, for all $i \in [1, n]$. $X_a = \cup_{p \in a} X_p$ is the set of variables available to an interaction α. G_a is the set of boolean guards associated to α. F_a is the set of the update functions associated to α and defined over X_a.

In the remainder of the paper, we may denote the interaction (a, X_a, G_a, F_a) by its set of ports a. An interaction a is enabled for a valuation v_a of X_a if and only if, for all $i \in [1, n]$, the port in $a \cap P_i$ is enabled in B_i and $G_a(v_a) =$ *True*. That is, an interaction is enabled if each port that is participating in this interaction is enabled and the guard evaluates to *True*.

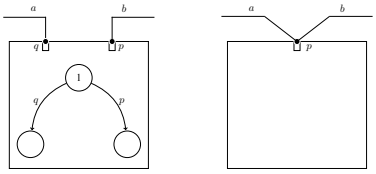
We denote by $comp(a)$ the set of components that have ports participating in a. $comp(a)$ is formally defined as:

$$comp(a) = \{B_i | i \in [1, n], P_i \cap a \neq \emptyset\} \quad (4)$$

Two interactions are conflicting at a given state of the system if both are enabled, but it is not possible to execute both from that state (i.e., the execution of one of them disables the other). In fact the enabledness of interactions only indirectly depends on the current state, through the enabledness of the participating ports. In systems without priorities, two interactions a and b may conflict only if they involve a shared component. In Figure 1a, the conflict comes from the fact that a and b involve two ports p and q of the same component labelling two transitions enabled from the same location. When reaching location, the component can execute either transition labelled by p or the one labelled by q but not both. This implies that when a and b are enabled, only one of them should execute. Figure 1b depicts a special case of conflict where interactions a and b share a common port p. Update functions of a and b may update variables exported by port p. This implies that when a and b are enabled, only one of them should execute.

***Definition** 4:* (Conflicting interactions).

Let $\gamma(B_1, ..., B_n)$ be a BIP model. We say that two interactions a and b of γ are conflicting, iff, there exists an atomic component $B_i \in comp(a) \cap comp(b)$ that has two transitions $\tau_a = (l, p, l'_1)$ and $\tau_b = (l, q, l'_2)$ from the same control location l such that $p \in a$ and $q \in b$. We denoted the conflict between a and b by $a \# b$. If a and b are not conflicting we say that they are independent. The system is conflict-free if all interactions are pairwise-independent.



(a) (b)

Figure 1. Conflicting interactions

Priorities are defined in order to reduce non-determinism in the system, that is, they are used to filter interactions among the enabled ones.

***Definition** 5:* (Priority in BIP).

Given a set γ of interactions defined over a set of components $\{B_1, ..., B_n\}$, we define a priority as a relation $\pi \subseteq \mathbb{B}^Q \times \gamma \times \gamma$, where $\mathbb{B}$ is the set of booleans and Q the set of states, such that for all $(C, a, a') \in \pi$, C depends only on data variables that are associated with ports of interactions a or a' and $\forall q \in Q$, $\pi_q = \{(a, a') \in \gamma \times \gamma \mid C(q) \wedge (C, a, a') \in \pi\}$ is a partial order. We say that a has less priority than a' whenever the predicate C holds.

The predicate C depends on the data variables exported by the participants in some interactions, allowing the corresponding priority rule to be dynamically enabled. A static priority is expressed by having $C =$ *True* for all $(C, a, a') \in \pi$.

A composite component is built from a set of n components $\{B_i = (L_i, P_i, X_i, C_i, T_i, \{tpc_l\}_{l \in L_i})\}_{i=1}^n$ such that their respective sets of places, ports, clocks, and discrete variables are pairwise disjoint; i.e., for any two $i \neq j$ from $[1, n]$, we have $L_i \cap L_j = \emptyset$, $P_i \cap P_j = \emptyset$, $C_i \cap C_j = \emptyset$, and $X_i \cap X_j = \emptyset$.

***Definition** 6:* (Composite Component with interactions). Let γ be a set of interactions. We denote by $B \stackrel{def}{=} \gamma(B_1, ..., B_n)$ the composite component obtained by applying γ to the set of components $\{B_i\}_{i=1}^n$. It is defined by the component $B = (L, P, X, C, T, \{tpc_l\}_{l \in L})$ as follows: $L = L_1 \times ... \times L_n$ is the set of locations, $P = \bigcup_{i=1}^n P_i$ is the set of ports, $X = \bigcup_{i=1}^n X_i$ is the set of variables, $C = \bigcup_{i=1}^n C_i$ is the set of clocks, Let $\alpha = (a, X_a, G_a, F_a) \in \gamma$ be an interaction. We denote $I_a = i \in [1, n] | B_i \in comp(a)$. A transition $\tau = (l, a, g_\tau, f_\tau, R, l')$ from $l = (l_1, ..., l_n)$ to $l' = (l'_1, ..., l'_n)$ ($l = l'$ if $i \notin I_a$) is in T if its projection $\tau_i = (l_i, p_i, g_{\tau_i}, f_{\tau_i}, R_i, l'_i)$ is a transition of B_i for all $i \in I_a$, where $g_{\tau i}$ and $f_{\tau i}$ are such that :

- $g_\tau = G_a \wedge \bigwedge_{i \in I_a} g_{\tau_i}$,
- $f_\tau = f_{\tau_1} \circ ... \circ f_{\tau_n} \circ F_a$ where f_{τ_i} is the identity function, for $i \notin I_a$ (Notice that functions f_{τ_i} modify disjoint sets of variables, hence can be composed in any order),
- $R = \bigcup_{i=I_a} R_i$.

For a control location $l = (l_1, ..., l_n) \in L$, the time progress condition is $tpc_l = \bigwedge_{i \in [1,n]} tpc_{l_i}$.

***Definition** 7:* (Composite Component with priorities). Let $B^\gamma = \gamma(B_1, ..., B_n)$ be the composite component obtained by applying γ to the set of components $\{B_i\}_{i=1}^n$. And let L be their set of locations and π the set of priority rules π_l for $l \in L$. We denote by $B^{\pi\gamma} = \pi\gamma(B_1, ..., B_n)$ the composite component obtained by applying priorities to the set of components B^γ: Given two interactions $a, a' \in \gamma$, with corresponding transitions $\tau^\gamma = (l, a, g_{\tau^\gamma}, f_{\tau^\gamma}, R_a, l')$ and $\tau'^\gamma = (l, a', g_{\tau'^\gamma}, f_{\tau'^\gamma}, r_{a'}, l'')$ in B^γ, such that $\exists (C, a, a') \in \pi$ (i.e. a has less priority than a' in the current location l), the corresponding transitions in $B^{\pi\gamma}$ are:

- $\tau^{\pi\gamma} = (l, a, g_{\tau^{\pi\gamma}}, f_{\tau^\gamma}, R_a, l')$, with $g_{\tau^{\pi\gamma}} = g_{\tau^\gamma} \wedge \neg g_{\tau'^\gamma} \wedge C$,
- and $\tau'^{\pi\gamma} = (l, a', g_{\tau'^{\pi\gamma}}, f_{\tau'^\gamma}, R_{a'}, l'')$, with $g_{\tau'^{\pi\gamma}} = g_{\tau'^\gamma}$.

The execution of interactions, taking into account priority rules and execution of local code of components are orchestrated by a sequential *scheduler*. Conflicts that are not resolved by priority rules are resolved by this scheduler; it randomly chooses one of simultaneously enabled interactions.

Components that are not composite, i.e. specified directly as LTS in the model, are called in the remainder of the article atomic components.

C. SR-BIP model

The SR-BIP models are designed, for untimed [8], [12] and timed [10] BIP, to automatically derive distributed implementations. They are intended to be implementable using basic message-passing primitives. The execution of a distributed process is a sequence of actions that are either message emission, message reception or internal computation. Consequently the SR-BIP model includes three types of ports: send-ports, receive-ports and unary-ports. Unary-ports correspond to internal computation. They can only appear in unary interactions, that is interactions involving only one component. Send and receive ports appear only in message-passing interactions (called send/receive interactions). Such an interaction has no guard, and the update function copies variables exported by the send-port to variables exported by the receive-port. In a canonic message-passing environment, each send action has a well-defined recipient. Therefore, it is required in SR-BIP models that each send-port participates in exactly one send/receive interaction. The latter ensures that for each send-port there is a unique corresponding receive-port.

An SR-BIP model is an RT-BIP model which contains components glued by send/receive interactions. A send/receive interaction is composed of one send-port and one or more receive-ports depending on the data transfer direction.

Since concurrency and distribution introduced to RT-BIP model can not be handled by sequential single scheduler, SR-BIP model handles interactions in dedicated schedulers, and resolves conflicts through conflict resolution protocol. It complies with a 3-layer architecture consisting of:

1) Components layer: The bottom layer consists of atomic components. Their interfaces are made of one send-port and one or more receive-ports. Components share their lists of enabled ports with the upper layer.

2) Schedulers: The second layer consists of a set of components each hosting a set of interactions. Conflicts between interactions included in the same component are resolved locally. And conflicts between interactions of different scheduler components are resolved using the third layer.

3) Conflict Resolution Protocol (CRP): This layer implements an algorithm based on the idea of message-count technique presented in [13]. It consists on counting the number of times that component participates in an interaction. Conflicts are resolved by ensuring that each participating number is used only once. Different implementations of the reservation protocol are presented in [8].

III. From High-level RT-BIP to TT physical model

In this section, we propose a generic framework for transforming an RT-BIP model into a TT physical model. We first detail subtleties of this transformation with respect to RT-BIP and SR-BIP. Then, we present the TT-BIP architecture that addresses these subtleties. Finally, we describe how to construct a correct TT-BIP model starting from a high-level RT-BIP model.

A. Subtleties of the transformation

In sequential models, interactions are executed atomically by a single scheduler. To the contrary, introducing TT settings (mainly decomposition into tasks and assuming TT communication mechanism) to this model requires the implementation to deal with more complex issues:

1) Decomposition into Tasks: Tasks (processes, threads, etc.) are building blocks for TT applications. In Design phase, designers have the choice to model a TT task using one or more BIP components. Thus depending on the user task mapping, tasks should appear in the derived physical model with respect to the initial high-level model.

2) Strong synchronization in BIP interactions Vs. message-passing: In order to respect TT communication setting, the derived physical model should handle intertask communication through dedicated RT-BIP component which stands for the TT communication system. The challenge is to switch from the high-level RT-BIP model, where multiparty synchronized interaction is a primitive, to the TT model, where inter-task communication is performed via a communication medium.

3) Resolving conflicts: In high-level RT-BIP model, conflicts are handled by the single scheduler. TT communication components in the derived model must ensure that execution of conflicting interactions is mutually exclusive.

We address the first issue, by initiating composite task components that encompass atomic components mapped to the same task. The second problem is addressed by breaking the atomicity of execution of interactions, so that a task can execute unobservable actions to notify TT communication component about their states, and then execute the corresponding interaction. Communication between tasks and TT communication component is send/receive interactions. Resolving conflicts leads us to use solution proposed in SR-BIP model, which consists in instantiating a BIP component that implements the algorithm proposed in [13]. The latter uses message counts to ensure synchronization and reduces the conflict resolution problem to dining or drinking philosophers [14].

B. TT-BIP Architecture

In this subsection, we present the TT-BIP architecture that combines elements of RT-BIP, SR-BIP and the TT paradigm to address previously mentioned issues. This architecture is based on *tasks* (processes, threads, etc.) as one of its building blocks. A task can be mapped on one or more atomic

components. Inter-task communication is handled by dedicated components called Time-Triggered Communication Components (TTCC) standing for schedulers in RT-BIP and SR-BIP models. Conflicts between different TTCC components, are resolved through CRP components of SR-BIP model.

An RT-BIP model B^{TT} complies with the TT-BIP architecture if it consists of three basic entities: Tasks, TTCC and CRP components, organized by the following abstract grammar:

$$
\begin{aligned}
TT\text{-}BIP\text{-}Model &::= Task^{+} \,.\, TTCC^{+} \,.\, CRP \,.\, S/R\text{-}connector^{+} \\
Task &::= atomic\text{-}component^{+} \,. \\
&\quad\; atomic\text{-}talking\text{-}component^{+} \,.\, connectors^{+} \\
TTCC &::= TTCC^{NC} \mid TTCC^{C}
\end{aligned}
$$

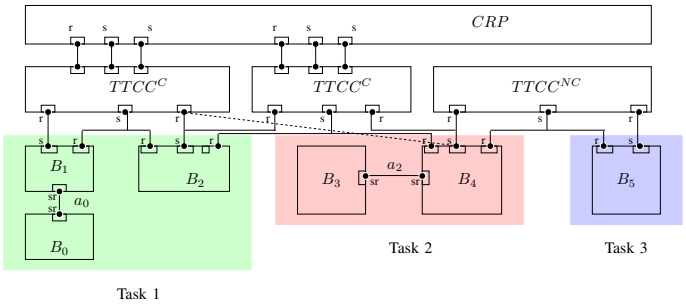


Figure 2. TT-BIP model

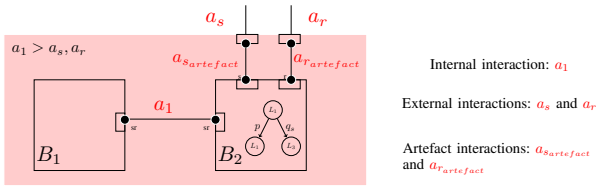
Task components (resp. TTCC components) and TTCCs (resp. CRP components) communicate with each other through message-passing, i.e., send/receive interactions. This latter is a set of one send port and one or more receive ports. Communication between components inside a task are classic multi-party RT-BIP interactions (see. Figure 2).

***Definition** 8:* We say that $B^{TT} = \pi^{TT}\gamma^{TT}(B_1^{TT}, ..., B_n^{TT})$ is a TT-BIP model iff we can partition the set of its interactions in two sets A_I and A_E that are respectively sets of *internal* and *external* interactions, corresponding to intra-task and inter-task interactions, such that:

- Each interaction $a \in A_I$, is a classic multi-party interaction presented by classic RT-BIP connectors relating atomic components inside one task component,
- Ports of Task, TTCC and CRP components can be partitioned into three sets P_u, P_s and P_r that are respectively the set of unary ports, send ports and receive ports. Each interaction $a \in A_E$, is either (1) a send/receive interaction with $a = s, r_1, ..., r_k$, $s \in P_s$, $r_1, ..., r_k \in P_r$ or, (2) a unary interaction $a = p$ with $p \in P_u$,
- The order of execution of transitions labelled by send or receive-ports depends on the nature of component (task, TTCC or CRP component). In task (resp. TTCC and CRP) components, each one or two successive transition(s) labelled by send-port(s) (rep. receive-ports) should be acknowledged by a transition labelled by receive-port (resp. send-port),
- If s is a port in P_s, then there exists one and only one send/receive interaction $a \in \gamma^{TT}$ with $a = (s, r_1, ..., r_k)$ and all ports $r_1, ..., r_k$ are receive-ports. We say that $r_1, ..., r_k$ are receive-ports of s,
- If $a = (s, r_1, ..., r_k)$ is a send/receive interaction in γ^{TT} and s is enabled at some global state of B^{TT}, then all its receive-ports $r_1, ..., r_k$ are also enabled at that state,

The specificity of each constituent element of the TT-BIP architecture is detailed below.

1) Task components: A task component is a composite component consisting of one or more atomic components related to each other using connectors. Atomic components within a task which export their send and receive ports to the task interface are called *atomic-talking-components* (ATC). These latter can only communicate with a TTCC components (through the task component interface). TTCC component interferes only in *external interactions* referring to inter-task interactions A_E. These *external interactions* are presented locally (in the task component) by *artefact interactions*. We denote the set of these artefact interactions by $A_{E_{artefact}}$. The remaining components in a task, do not export their ports. They can only participate in *internal interactions* A_I (see Figure 3).



Task

Figure 3. Example of a task component

In task component, conflicts between internal and artefact interactions can be resolved by local priority rules. In such cases, execution of internal interactions are privileged to external ones, that is, internal interactions have more priority than artefact ones (see. Figure 3). Through these priority rules, all possible conflict cases are handled. If the conflict can be resolved locally, the priority rule $a \in A_I > a' \in A_{artefact}$ can be used. Otherwise, the conflict will be handled by TTCC and CRP components.

Each ATC component in the task component has one or more send and receive ports and zero or more standard ports. It contains in addition to observable states and transitions, some partial states and unobservable transitions. An unobservable transition always leads to a partial state. Transitions allowing communication with the TTCC could be seen as a non atomic transition. It starts first by a non visible transition labelled by the send port. It allows to send an offer (i.e., information about active ports) to TTCC component. This transition is followed then by a visible transition labelled by the receive port indicating the completion of the communication. These two transitions are separated by a partial-state location standing for a busy state of the component where it is waiting for a notification from the TTCC via the receive port (cf. Figure 4).

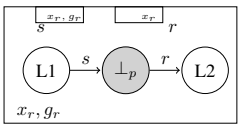
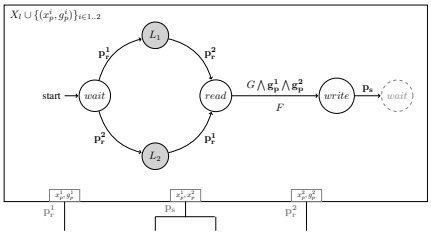


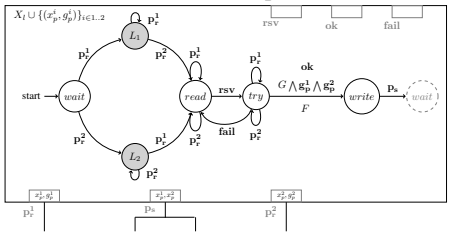
Figure 4. ATC Communication pattern

2) TTCC Component: Each TTCC component represents one interaction $a \in \gamma$ with $a = (a, G_a, F_a)$, we denote by n the number of components related to TTCC. It is inspired from schedulers of the SR-BIP models implemented for distributed implementations in BIP [9] and RT-BIP [10]. The TTCC component is designed to model the behaviour of

the communication system in the TT paradigm in order to execute an interaction a. There are two different types of TTCC components: conflicting and non-conflicting, denoted respectively $TTCC^C$ and $TTCC^{NC}$. TTCC component behaviour is made of three steps; (1) the component reads variables and guards from task components, (2) based on these received guards and the guard of the interaction a, the TTCC component takes a decision by either executing the interaction upon synchronization (i.e., conjunction of read guards evaluates to *True*) if a is a non-conflicting interaction or soliciting the CRP component to find out if the conflicting interaction a can be executed and (3) finally it writes on appropriate task components by sending a notification. Figure 5a presents an example of $TTCC^{NC}$ and Figure 5b presents an example of $TTCC^C$ for $n = 2$.



(a) $TTCC^{NC}$ component, n=2



(b) $TTCC^C$ component, n=2

Figure 5. Conflicting and non conflicting TTCC components

The behaviour of the TTCC components is described as a timed automata. It depends on whether the TTCC handling an interaction a is conflicting ($TTCC^C$) or not ($TTCC^{NC}$).

The set of variables of TTCC can be partitioned into two classes. The first class consists in variables updated whenever an offer from ATC components B_i participating in the interaction a is received. They consist of the following: guard variable g_p^i and a local copy of the variables X_p^i for each port p involved in the interaction a, a time progress condition variable tpc_i and a participation number variable n_i for each ATC component B_i of tasks participating in a. The second class consists in variables updated whenever interaction a is scheduled which are: execution date variable t_{ex_a}, that stores the last execution date of interaction a and, an execution date variable t_{ex_i} for each component B_i participating in a.

In case of a non conflicting TTCC ($TTCC^{NC}$), the set of places contains the followings: For each ATC component B_i within a task involved in the interaction a, we include waiting places: $wait$ and the set $L_\perp$ with $|L_\perp| = \sum_{k=0}^{n-2} \prod_{j=0}^{k} (n-j)$. $L_\perp$ locations present states where component is waiting for reception of other tasks guards and variables. Each place $l_\perp \in L_\perp$ that is reached by a transition receiving offer from B_i, has a time progress condition defined by the variable tpc_i. We include a writing place $write$ that allows notification of tasks (and then ATC components) participating in a. The time progress condition of $write$ is *False*. In case of a conflicting TTCC ($TTCC^C$), the set of states includes the same states of a non conflicting TTCC, with an additional trying state try_a.

The set of ports of a non conflicting TTCC consists of the following: For each ATC component B_i within a task involved in the interaction a,TTCC includes a receive-port p_{r_i}, to receive offers. Each port p_{r_i} is associated with the variables g_p^i and X_p^i for each port p of B_i as well as the variable tpc_i and n_i of B_i. The set of ports of TTCC includes a send-port p_s, which exports the set of variables $\cup_{p \in a} X_p \cup \{t_{ex_i}\}$ where i is the index of ATC component B_i that exports port p. TTCC includes also a unary port a allowing the execution of the interaction a. In case of a conflicting TTCC ($TTCC^C$), the set of ports includes the same ports of a non conflicting TTCC except the unary one, with additional ports; rsv_a associated with the set of variables n_i of components $B_i \in comp(a)$.

Transitions of a non conflicting TTCC are as following: In order to receive data from task components; we include for each $\{l, l'\} \subseteq \{\{wait, read\} \cup L_\perp\}$ and each $i \in [1..n]$ a transition $\tau_{receive_i} = (l, p_r^i, True, Identity, l')$. This transition guard is default to *True*. In order to execute the interaction a we include the Transition leading from state $read$ to state $write$, where $\tau_a = (read, p_u, G_a \bigwedge (\bigwedge_{i=1}^{n} g_p^i), F_a, write)$ where p_u is a unary port, G_a is a guard on variables of the set X_l, and F_a is a function on X_l. To notify task components after executing the interaction a, we include the transition $\tau_{write} = (write, p_s, True, Identity, wait)$.
In case of a conflicting TTCC ($TTCC^C$), the set of transitions includes in addition to transitions $\tau_{receive}$ and τ_{write} of non conflicting TTCC, the following transitions: To each place $l_\perp \in L_\perp$ reached by transition labelled by port p_{r_i} we include a loop transition $\tau_{loop_i} = (l_\perp, p_r^i, True, Identity, l_\perp)$. Before executing interaction a, we include the transition that will solicit the CRP component from the state $read$, where $\tau_{rsv} = (read, rsv, True, Identity, try)$. We also include loop transitions on place $read$, allowing reception from receive ports, such that for all $i \in [1, n]$, $\tau'_{loop_i} = (read, p_r^i, True, identity, read)$. From try location, the first possible transition, depending on the response of the CRP component, $\tau_{ok} = (try, ok, G_a \bigwedge (\bigwedge_{i=1}^{n} g_p^i), F_a, write)$, where G_a is a guard on variables of the set X_l, and F_a is a function on X_l. This transition will execute the interaction a. We also include loop transitions on place try, allowing reception from receive ports, such that for all $i \in [1, n]$, $\tau''_{loop_i} = (try, p_r^i, True, identity, try)$. The second possible transition from state try is $\tau_{fail} = (try, fail, True, Identity, read)$, in that case, a is not allowed to execute.

3) Conflict Resolution Protocol Component: The conflict resolution protocol (CRP) accommodates the algorithm proposed in [13]. It uses message counts to ensure synchronization and reduces the conflict resolution problem to dining or drinking philosophers [14]. Its main role is to check the freshness of requests received for an interaction, that is, to check that no conflicting interactions has been already executed using the same request. In each request, an interaction sends the participation numbers of its components, i.e., number of interactions each ATC component has participated in. This ensures that two conflicting interactions cannot execute with the same request. Mutual exclusion is ensured using participation

 ISBN: 978-1-61208-438-1

numbers. To this end, the conflict resolution protocol keeps the last participation number N_i of each component B_i and compares it with the participation number n_i provided along with the reservation request from TTCC components. If each participation number from the request is greater than the one recorded by the conflict resolution protocol ($n_i > N_i$), the interaction is then granted to execute and N_i is updated to n_i. Otherwise, the interaction execution is disallowed. Figure 6 presents the places, transitions, variables, guards and update functions involved in handling an interaction a with two participating components B_1 and B_2. Whenever a reservation for executing a arrives, this token moves from place $wait_a$ to place $receive_a$. From this place, if the guard of the transition labelled by ok_a is *True*according to the current values of N_i variable and freshly received n_i variables, the transition can take place. The transition labelled by $fail_a$ is always possible.

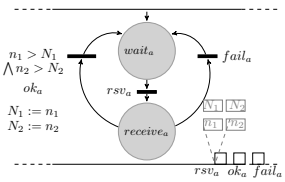


Figure 6. Fragment of the CRP component

C. From High-level RT-BIP model to TT-BIP model

In this section, we present the model transformation from a RT-BIP model $B \stackrel{def}{=} \pi\gamma(B_1, ..., B_n)$ into an equivalent TT-BIP model $B^{TT} = \pi^{TT}\gamma^{TT}(B_1^{TT}, ..., B_n^{TT}, TTCC_1, ..., TTCC_m, CRP)$.

This transformation is parametrized by a user-defined task mapping which consists in associating to each task T_k a group of atomic components of the model B. We denote by $\mathcal{B}$ the set of atomic components of model B. We assume, we have $K \leq n$ tasks and we denote by $\mathcal{T} = \{T_k\}_{k \in K}$ the Task set, such that $\mathcal{T}$ is a partition of $\mathcal{B}$: where for all $j, k \in K$ and $j \neq k, T_j \cap T_k = \emptyset$. For all $k \in K$ we have $T_k = \{B_i\}_{i \in I_k}, I_k \subseteq K$ such that $\bigcup_{k \in K} I_k = K$.

The transformation process is performed in two steps as shown in Figure 7. First, an analysis phase allows definition of set of components and connectors to be transformed depending on the task mapping. Then, the RT-BIP model is transformed into a TT-BIP model where only inter-task interactions are replaced by TTCC components and intra-task interactions remain intact. Components mapped to the same task are gathered in a composite task component.

We consider that the original RT-BIP model consists only of atomic components and flat connectors. Moreover, each connector defines one and only one interaction. Indeed, these assumptions do not impose any restrictions on the original model, since we can apply the transformations "Component flattening" and "Connector flattening" of previous research work in [15].

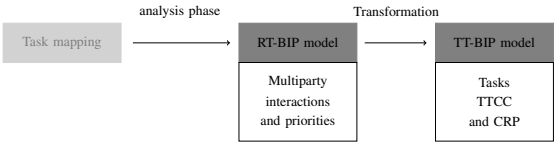


Figure 7. Transformation from RT-BIP to TT-BIP model

D. Analysis Phase

Starting from a high-level RT-BIP model and a user task mapping, a TT-BIP model can be derived. Thus, we have first to identify internal and external interactions as well as ATC components. Conflicts between external interactions are held by CRP components. But conflicts between an internal and external interactions should be also resolved. We distinguish between three cases of a conflict between an internal and external interaction; (1) conflict not resolved in the initial model by a priority rule, (2) conflict initially resolved by a static priority rule and (3) a conflict initially resolved by a dynamic priority rule. These three cases should be taken into account. In the first case, the conflict can be resolved locally in the task by imposing a priority rule that privileges the internal interaction (see Figure 3). In the second case, the conflict is already resolved and the initial priority rule is kept in the derived model. In the third case, task component can't resolve the conflict locally, since it doesn't have access to the global state of the model and thus can't evaluate the dynamic priority rule. In that case, the internal conflicting interaction is exported to be held by a dedicated TTCC. This interaction is thus being an external interaction. And the conflict is being a conflict between two external interactions. It can now be resolved by the CRP component initially intended to handle such a conflict.

The algorithm of the analysis phase initialises sets of internal, external and ATC components (A'_I, A'_E and B'^{ATC}) based on the task mapping. And then, it defines new sets (A_I, A_E and B^{ATC}) depending on conflicts between internal and external interactions and existing priority rules related to this conflict. These obtained sets are inputs for the transformation process.

A'_I, A'_E and B'^{ATC} sets are defined depending on the user task mapping as follows:

- External interactions: A'_E set is defined as the set of interactions in the participant components of which we can find at least two different atomic components belonging to two different tasks.
  Formally, $A'_E = \{a \in \gamma \mid \exists B_1, B_2 \in comp(a), T_1, T_2 \in \mathcal{T} : B_1 \in T_1, B_2 \in T_2, T_1 \neq T_2\}$.
- Internal interactions: A'_I set in defined as the set of interactions in the participant components of which we can find only atomic components belonging to the same task. $A'_I = \gamma \setminus A'_E$.
- Atomic talking components: $\mathcal{B}'^{ATC}$ set is initialised to the set of atomic components in $\mathcal{B}$ that are concerned by external interactions (i.e., are related to other atomic components belonging to a different task). We define $\mathcal{B}'^{ATC} = \{B \in \mathcal{B} | A'_E \cap P_B \neq \emptyset\}$, where P_B is the port set of the component B.

After defining initial sets of internal interactions, external interactions and ATC components, the final sets A_E, A_I and $\mathcal{B}^{ATC}$ are defined depending on existing priority rules following Rule 1.

***Rule** 1:* For each $a \in A'_I$, and, $a' \in A'_E$ such that $\exists q \in Q$, $(C(Q), a, a') \in \pi_Q$, $A_E = A'_E \cup \{a\}$, $A_I = \gamma \setminus A_E$, $\mathcal{B}^{ATC} = \{B \in \mathcal{B} | A_E \cap P_B \neq \emptyset\}$.

E. Transformation rules

Starting from an RT-BIP model $B \stackrel{def}{=} \pi\gamma(B_1, ..., B_n)$, we first apply a variation of the transformation from RT-BIP

to SR-BIP [8] which transforms original atomic components defined in the RT-BIP model by send/receive components, handles initial interactions using schedulers and adds a conflict resolution protocol component for conflict resolution. Then, we instantiate components associated to the same task in one composite task component following Rule 4.

In our case, we need to implement intertask communication using dedicating components instead of BIP connectors. Thus, we propose here to apply the send/receive transformation only to the subset of components and connectors dedicated to inter-task communication in the initial model. Thus, only ATC components are transformed in send/receive atomic components following Rule 2, and only inter-task connectors referring to external interactions A_E are replaced by TTCC components and the CRP component following Rule 3.

1) ATC component transformation: Every atomic component can define a set of local clocks. They can be reset at any time and are involved in timing constraints and time progress conditions. TTCC component makes use only of the common time scale presented by a global clock C^gwhich is initialized to 0 and is never reset, and measures the absolute time elapsed since the system started executing.

Since TTCC receives timing constraints and time progress conditions from different ATC components, it would be better if these latter transforms their sent data using the global clock. Therefore, we follow the approach of [6]: for each clock c of an atomic component B, we introduce a variable ρ_c that stores the absolute time of the last reset of c. This variable is updated whenever the component executes a transition resetting clock c. In fact, when the TTCC executes an interaction a, it stores its execution date in a variable t_{ex_a}. The value of this variable is then sent to participating components (during the notification). Each participating component executes then the corresponding transition according to the received notification, and updates each variable ρ_c to t_{ex_a} if the transition resets clock c in the original model. Notice that the value of c can be computed from the current value of C^g and ρ_c by using the equality $c = C^g - \rho_c$. Using ρ_c, any timing constraint tc involved in a component B can be expressed using the clock C^g instead of clocks C. We can transform tc as follows:

$$tc = \bigvee_{k=1}^{m} \bigwedge_{c \in C} l_c^k \leqslant c \leqslant u_c^k = \bigvee_{k=1}^{m} \bigwedge_{c \in C} l_c^k + \rho_c \leqslant C^g \leqslant u_c^k + \rho_c \quad (5)$$

Similarly, any time progress condition tpc involved in B is transformed using the clock C^g as follows:

$$tpc = \bigvee_{c \in C} c \leqslant u_c = \bigvee_{c \in C} C^g \leqslant u_c + \rho_c \quad (6)$$

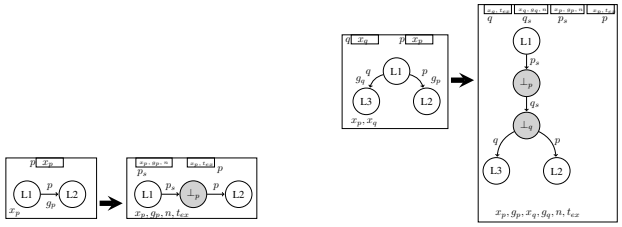
We transform then an ATC component $B^{ATC} = (L, P, X, C, T, \{tpc_l\}_{l \in L})$ of a RT-BIP model into a TT ATC component $B^{TT} = (L^{TT}, P^{TT}, X^{TT}, C^{TT}, T^{TT}, \{tpc_l^{TT}\}_{l \in L})$ that is capable of communicating with $TTCC$ component(s). We denote by a the initial interaction between two or more ATC components and by p the port of B^{ATC} such that $p \in a$. Instead of the initial connector linking these atomic components, communication is performed through added $TTCC$ component. Thus, each atomic component will be connected to $TTCC$ component using send/receive connectors.

***Rule** 2:* In each ATC component $B^{ATC} \in \mathcal{B}^{ATC}$, for each port $p \in P \cap A_E$, each transition labelled by the port p will be duplicated in two transitions with an intermediate partial state $\perp_p$. A new added send-port p_s and the port p (Receive port) are the successive labels of the two new transitions. The transition labelled by p_s is an unobservable transition. Formally, B^{TT} is obtained from B as follows:

- $L^{TT} = L \cup L_\perp$, where $L_\perp = \{\perp_{l,p} \ | \exists \tau = (l, p, g_\tau, f_\tau, r, l') \in T, a \in A_E, p \in P \cap a\}$,
- $P^{TT} = P \cup P_s$, where $P_s = \{p_s | p \in P \cap A_E\}$,
- $X^{TT} = X \cup \{g_p\}_{p \in P \cap A_E} \cup \{nb, \rho_c, t_{ex_a}\}$, where nb is the number of participation of B^{TT},
- $C^{TT} = C, C^g$,
- $T^{TT} = T \backslash \{\tau_p\}_{p \in P \cap A_E} \cup \{\tau_{p_s}, \tau'_p\}_{p \in P \cap A_E}$ where $p \in P \cap A_E$, $\tau_p \in T$ and $\tau_{p_s}, \tau'_p \in T^{TT}$ such that $\tau_p = (l, p, g_{\tau_p}, f_{\tau_p}, l')$, $\tau_{p_s} = (l, p_s, True, g_p = g_{\tau_p}, L_\perp)$ and $\tau'_p = (L_\perp, p, True, f_{\tau'_p}, l')$. $f_{\tau'_p}$ executes f_{τ_p} and increments nb.
- For places $L_\perp$, the time progress condition is $tpc_{L_\perp}^{TT} = True$.

We denote the set of transformed ATC components by ${\mathcal{B}_{\mathcal{ATC}}}^{TT}$.

Figure 8 illustrates this transformation for the cases of conflicting and non conflicting interactions.



(a) Non conflicting interaction (b) Conflicting interaction

Figure 8. Atomic Component transformation

2) Connector transformation: Each connector presenting an inter-task communication will be replaced by a TTCC component as described by the following rule:

***Rule** 3:* Let C_k be a connector defining an external communication between k components of different tasks which handles an external interaction $(a, X_a, G_a, F_a) \in A_E$. We replace C_k by a $TTCC_k$ component and a set of corresponding connectors, as follows::

- if a is a conflicting interaction (i.e., $\exists a' \in A_E$ such that $a \# a'$), $TTCC_k$ is an instance of $TTCC^C$ components, else $TTCC_k$ is an instance of $TTCC^{NC}$ component.
- In both cases, the set of local variables of $TTCC_k$ is $X_l = X_a \cup \{nb, t_{ex}\}$, and in the transition executing the interaction a, we have the guard $G = G_a \bigwedge (\bigwedge_{i=1}^{k}) g_p^i$ and the function $F = F_a$.
- k binary and one k+1$-ary$ send/receive connectors are instantiated to connect $TTCC$ to ATC components,
- A CRP component is then instantiated and connected to $TTCC^C$ components via three binary send/receive connectors each.

3) Task component instantiation: Task components are created according to the task mapping. Each transformed ATC component exports its send and receive ports to the task component interface. Each exported port will be referenced by an artefact external interaction $a_{artefact}$ that presents locally the external interaction involving the task. Conflicts between internal and artefact interactions that are not initially resolved by priority rules, are resolved by a static added rule that favours the internal interaction.

***Rule** 4:* For each task $T_k = \{B_i\}_{i \in I_k} \in \mathcal{T}$, we instantiate a composite component T_k^{TT} where:

- The set of exported ports of T_k^{TT} is $P_T^{TT} = (P_s \cap \bigcup_{B_i^{TT} \in T_k^{TT} \cap \mathcal{B}_{\mathcal{ATC}}{}^{TT}} P_{B_i^{TT}}) \cup (P_r \cap \bigcup_{B_i^{TT} \in T_k^{TT} \cap \mathcal{B}_{\mathcal{ATC}}{}^{TT}} P_{B_i^{TT}})$,
- for each interaction $a \in A_I$ in conflict with an interaction $b \in A_E$, we add the priority rule $\pi = (\mathit{True}, b, a)$.

In this section, we proposed TT-BIP architecture as a TT physical model that allows inter-task interactions only through a communication media (TTCC components) in order to comply with the TT paradigm. We also defined a transformation process that derives correctly a TT-BIP model from a high-level RT-BIP model. The obtained TT-BIP model is intended to be translated into the language specific to target TT platform. Since each operating system (OS) offers specific services (communication, scheduling policy, etc.), it is advantageous to define a mapping between these offered services and their corresponding elements in the TT-BIP model. To be able to do such a mapping, we consider the decomposition of the TTCC component into an equivalent set of atomic components each describing a step of its behaviour: components allowing individual acquisition of offers, components to aggregate them, and component executing the interaction. We believe that some parts of TTCC subcomponents can be mapped into OS communication services. Priorities in the TT-BIP model (including added ones in task component) are priorities on interactions. In simulation level they can be resolved by RT-BIP sequential scheduler. In order to be handled in implementation, these priority rules should be translated either into predicates on transitions following [9], or, into priorities on tasks, which could parametrize the scheduler of the target platform. This adequation between subset of TTCC (resp. priority rules) and OS communication services (resp. schedulers) will be the object of a future publication.

IV. Case Study

In this section, we present a simple but representative case study which consists in a Flight Simulator (FS) application. We have first modelled the FS application in RT-BIP starting from FS Modelica model. Then, we applied manually the transformation to derive the equivalent TT-BIP model. The simulation of three models reveals that their outputs are similar.

The initial RT-BIP model, consists of a set of six communicating components (see. Figure 9): autopilot, fly-by-wire, route, servo, simulator and sensor. The autopilot models the pilot commands in function of the flight state. It has four main functionalities: flight state reception from sensor component, execution of the route planner, execution of fly-by-wire and sending command to servo component. The route component computes distance to current waypoint and change route towards next waypoint if necessary. It operates in low frequency: every 15 seconds. The fly-by-wire component allows course correction by setting roll attitude and ailerons and elevator. It operates in high frequency: every 5 seconds. The servo refers to the autopilot's actuation on plane's flight control surfaces. Servo component receives command from autopilot component and transfers it to simulator component. The Flight simulator simulates flight dynamics computation of plane and wing tips position based on received commands (i.e., new values of roll, pitch and throttle). The sensor refers to the autopilot's perception of real world data. Sensor component receives data about flight state from simulator component and resend them to the autopilot. The sensor can add some noise, delay, etc., to mimic realistic data acquisitions. But in our example, we stand for copying the state computed by simulator component.

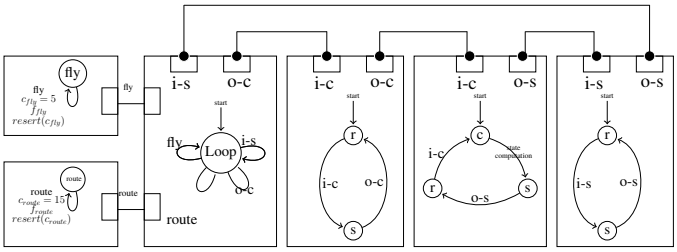


Figure 9. Initial Flightsim model

We apply the transformation and derive the TT-BIP model. The chosen task mapping is as follows; the first task compromises autopilot, route and fly-by-wire components. Servo, sim and sensor are each mapped on a different task. Thus, only connectors route and fly remain intact. Other connectors are inter-task connectors. They are transformed in TTCC components. The interactions connecting the autopilot component to the sensor and servo components are conflicting in the state "loop" of the autopilot. Thus, instantiated TTCC for these interactions are related to CRP component. Figure 10, shows the obtained model. Behaviours of TTCC and CRP components are not displayed for lack of space. Nonetheless, since all TTCC components are connecting exactly two tasks, their automata are strictly similar to those in Figure 5.

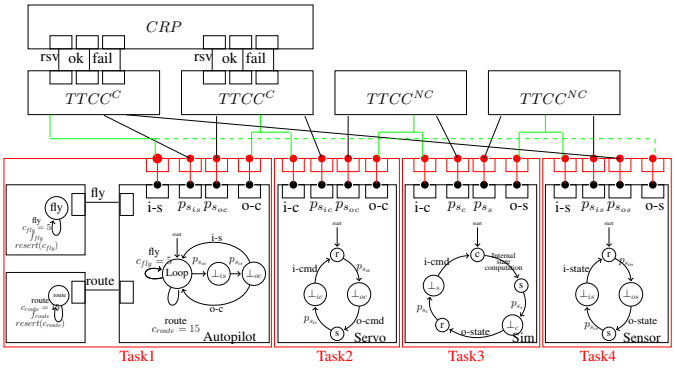


Figure 10. Final Flightsim model

In order to be able to compare the functionality of two models, we used BIP simulator that generates C++ code from the initial and the obtained model. Simulation of two generated codes, allowed us to visualize and compare the output signals. A band shows the trajectories of left and right wingtips and illustrates the roll movement that precedes the change in course at each waypoint, while the plane progressively reaches its desired altitude.

 ISBN: 978-1-61208-438-1

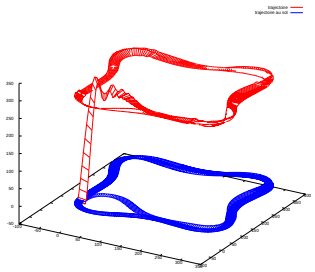


Figure 11. Trajectories of left and right wingtips

Figure 11 presents the simulation results of the initial and the derived models, for the waypoints (300,0,300), (300,300,300), (0,300,300) and (0,0,300). Visual inspection reveals that the output of the transformed model is strictly similar to that of the original model.

V. Related work

There have been a number of approaches exposing the relevant features of the underlying architectures at high level design tool.

Authors in [16] present a design framework based on UML diagrams for applications running on Time-Triggered Architecture (TTA). This approach doesn't support earlier architectural design phase and needs a backward mechanism for the generated code verification. It also doesn't target generic TT implementations since it assumes the underlying TT protocol to be The Flexray standard.

Authors of [17] and [18] present a method to reduce the gap between models used for timing analysis and for code generation. This method relies on AADL model transformations in order to lower automatically the abstraction level of models. However, this work did not rely on formal semantics.

Since BIP design flow is unique due to its single semantic framework used to support application modelling and to generate correct-by-construction code, many approaches tend to use it to translate high level models into physical models including architectural features. For instance, in [15], a distributed BIP model is generated from a high level one. In [19], a method is presented for generating a mixed hardware/software system model for many-core platforms from an application software and a mapping. These two approaches take advantages from BIP framework but they do not address the TT paradigm.

To the best of our knowledge, our approach is the first to address the problem of deriving progressively a TT physical model in a single host component-based language rooted in well defined semantics.

VI. Conclusion

In this paper, we proposed a chain of transformations that starts from an arbitrary abstract RT-BIP model, and a user-defined task mapping. The transformation process obtains a model that suits the TT-BIP architecture and consists of: (1) breaking atomicity of actions in ATC components by replacing strong synchronizations with asynchronous send/receive interactions, (2) inserting TTCC components that coordinate execution of inter-task interactions according to a user-defined task mapping, (3) extending the model with an algorithm for handling conflicts between TTCC and (4) adding local priority rules in task components for handling conflicts between inter-task and intra-task interactions. We have shown correctness of the final model by trace inclusion.

For future work, we plan to automate the transformation process in RT-BIP modelling environment. Furthermore, we are working on the tool allowing code generation for the target TT platform, that depends on the offered communication services of target operating system. For each specific target operating system, it translates an adapted version of the model.

References

[1] H. Kopetz, "The time-triggered approach to real-time system design," Predictably Dependable Computing Systems. Springer, 1995.

[2] ——, "Time-triggered real-time computing," Annual Reviews in Control, vol. 27, no. 1, 2003, pp. 3–13.

[3] W. Elmenreich, G. Bauer, and H. Kopetz, "The time-triggered paradigm," in Proceedings of the Workshop on Time-Triggered and Real-Time Communication, Manno, Switzerland, 2003.

[4] BIP2 Documentation, July 2012.

[5] T. Abdellatif, "Rigourous implementation of real-time systems," Ph.D. dissertation, UJF, 2012.

[6] T. Abdellatif, J. Combaz, and J. Sifakis, "Model-based implementation of real-time applications," May 2010, pp. 229–238.

[7] R. Alur and D. L. Dill, "A theory of timed automata," Theoretical computer science, vol. 126, no. 2, 1994, pp. 183–235.

[8] B. Bonakdarpour, M. Bozga, M. Jaber, J. Quilbeuf, and J. Sifakis, "From high-level component-based models to distributed implementations," in Proceedings of the tenth ACM international conference on Embedded software. ACM, 2010, pp. 209–218.

[9] J. Quilbeuf, "Distributed implementations of component-based systems with prioritized multiparty interactions. application to the bip framework." Ph.D. dissertation, Université de Grenoble, 2013.

[10] A. Triki, B. Bonakdarpour, J. Combaz, and S. Bensalem, "Automated conflict-free concurrent implementation of timed component-based models," in NASA Formal Methods. Springer, 2015, pp. 359–374.

[11] R. Alur and D. L. Dill, "A theory of timed automata," Theoretical computer science, vol. 126, no. 2, 1994, pp. 183–235.

[12] A. Basu, P. Bidinger, M. Bozga, and J. Sifakis, "Distributed semantics and implementation for systems with interaction and priority," in Formal Techniques for Networked and Distributed Systems–FORTE 2008. Springer, 2008, pp. 116–133.

[13] R. Bagrodia, "Process synchronization: Design and performance evaluation of distributed algorithms," Software Engineering, IEEE Transactions on, vol. 15, no. 9, 1989, pp. 1053–1065.

[14] K. M. Chandy and J. Misra, "The drinking philosophers problem," ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 6, no. 4, 1984, pp. 632–646.

[15] M. Bozga, M. Jaber, and J. Sifakis, "Source-to-source architecture transformation for performance optimization in bip," Industrial Informatics, IEEE Transactions on, vol. 6, no. 4, 2010, pp. 708–718.

[16] K. D. Nguyen, P. Thiagarajan, and W.-F. Wong, "A uml-based design framework for time-triggered applications," in Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International. IEEE, 2007, pp. 39–48.

[17] E. Borde, S. Rahmoun, F. Cadoret, L. Pautet, F. Singhoff, and P. Dissaux, "Architecture models refinement for fine grain timing analysis of embedded systems," in Rapid System Prototyping (RSP), 2014 25th IEEE International Symposium on. IEEE, 2014, pp. 44–50.

[18] F. Cadoret, E. Borde, S. Gardoll, and L. Pautet, "Design patterns for rule-based refinement of safety critical embedded systems models," in Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on. IEEE, 2012, pp. 67–76.

[19] P. Bourgos, "Rigorous design flow for programming manycore platforms," Ph.D. dissertation, Grenoble, 2013.

[20] R. Milner, Communication and Concurrency. Hertfordshire, UK, UK: Prentice Hall International (UK) Ltd., 1995.

 ISBN: 978-1-61208-438-1

APPENDIX
TRANSFORMATION CORRECTNESS PROOFS

In this section we prove the correctness of our TT-BIP model. First, we show that the obtained model is indeed a TT-BIP model. Second, in order to prove the correctness of our transformation, we consider these two steps in transforming a model B; the first step that transforms ATC components and inter-task connectors, and the second step that instantiates composite task components with priorities. We denote by B_{SR}^{TT} the output model of the first step and by B^{TT} the final model after applying the second step. We show that B weakly simulates B_{SR}^{TT}. And finally, we prove trace inclusion between B_{SR}^{TT} and B^{TT}, i.e., the second step of transformation preserves safety property.

A. Compliance with TT-BIP model

We need to show that receive ports of B^{TT} model will unconditionally become enabled whenever one of the corresponding send ports is enabled. Intuitively, this holds since communications between tasks and TTCC components, and between TTCC components and CRP component follow a request/acknowledgement pattern. Whenever a component sends a request (via a send port) it enables the receive port to receive acknowledgement.

In ATC components, we denote by $l_\perp$ in L^{TT} each place allowing sending offer to TTCC component. Note that $l_\perp$ precedes one $\perp_p$ place when only one intertask interaction is possible (e.g., the place L_1 in the right part of Figure 8a). It consists in all states allowing to send successive offers to TTCC components in case of conflicting intertask interactions (e.g., the places L_1 and $\perp_p$ in the right part of Figure 8b). $l_\perp$ places are called busy locations. We denote by $\perp_p^*$ each $\perp_p$ place from which no offer could be sent (e.g., the place L_p in the right part of Figure 8a and the place L_q in the right part of Figure 8b). We denote by l places that are neither $l_\perp$ nor $\perp_p^*$ places. In AC components, all places are denoted by l.

Lemma 1: Given an RT-BIP model $B = \pi\gamma(B_1, ..., B_n)$ and a task mapping $T = \{T_1, ..., T_k\}$, the model $B^{TT} = \pi^{TT}\gamma^{TT}(B_1^{TT}, ..., B_n^{TT}, TTCC_1, ..., TTCC_m, CRP)$ obtained by transformation of section III-C meets the properties of definition 8.

Proof:

The first four constraints of Definition 8 are trivially met by construction. We now prove that the fifth constraint also holds, i.e., whenever a send port is enabled, all its associated receive ports are enabled as well.

- Between a task component B_i^{TT} and a $TTCC_j^{NC}$ component, for all interactions involving a component B_i, we distinguish between four classes of states:
  - The first class contains all states between wait and Read states of $TTCC_j$ where this latter is enabling all its receive ports in all possible orders and B_i^{TT} is in a busy location $l_\perp$. This class presents waiting states. From that class, the only enabled send-port involved in an interaction with B_i^{TT} is the port p_{s_i} of B_i. By definition of the class, all associated receive ports are also enabled, and the send/receive interaction can take place to reach a state of the second class.
  - In the second class, the component B_i^{TT} is in a place $\perp_p^*$ that is not a busy location, and the TTCC component is in the Read place. From that configuration, there is no enabled send-port involved in an interaction with B_i^{TT}. The next class of states is reached when the TTCC executes a unary interaction a.
  - In the third class, the TTCC component is in the place write and B_i^{TT} component is still in $\perp_p^*$ place. The send-port p_s of TTCC is enabled. By definition of the class, the corresponding receive-port is enabled since component B_i^{TT} is in location $\perp_p^*$. Thus the send/receive interaction can take place either to reach back the first class of states or to reach the following class.
  - In the remaining class, the component B_i^{TT} is in place l and TTCC component is in Wait place. From this state only inta-task interactions could be enabled and no communication with TTCC component can be planned (no send-port is active).
- Between a task component B_i^{TT} and a $TTCC_j^C$ component, for all interactions involving the interaction a which is externally conflicting, we have almost the same four classes as non conflicting TTCC. $TTCC_j^C$ refers to CRP before executing a. It uses the current participation number of B_i for the execution of a and no other interaction is granted using the same participation number. Thus, write is the only active place, from which a notification could be sent to a component B_i.
- Between the TTCC component $TTCC_j^C$ and the conflict resolution protocol CRP, we consider the places try_a in the component $TTCC_j^C$, $wait_a$ and $treat_a$ in CRP component. If $TTCC_j^C$ in not in try_a place, and CRP in in the $wait_a$ place, only reservation request through port rsv_a is enabled. When the rsv_a request is sent, the place try_a and $treat_a$ become active. From this configuration, only send ports ok_a and $fail_a$ are enabled in the CRP, and the associated ports are also enabled in the TTCC (from try_a location).

■

This proof ensures that any component ready to perform a transition labelled by a send-port will not be blocked by waiting for the corresponding receive-ports.

B. Observational Equivalence between B and B_{SR}^{TT}

We denote by B the initial model and by B_{SR}^{TT} the resulting model of the step 1 of the transformation.

We show that B and B_{SR}^{TT} are observationally equivalent. The definition of observational equivalence between two transition systems $A = (Q_A, P_A \cup \{\beta\}, \underset{A}{\longrightarrow})$ and $B = (Q_B, P_B \cup \{\beta\}, \underset{B}{\longrightarrow})$ is based on the usual definition of weak bisimilarity [20], where β-transitions are considered unobservable.

Definition 9: (Weak Simulation A weak simulation over A and B, denoted $A \subset B$, is a relation $R \subset Q_A \times Q_B$, such that: $\forall (q, r) \in R$, $a \in P$: $q_A \overset{a}{\rightarrow}_A q' \implies \exists r'$: $(q', r') \in R \wedge r \overset{\beta^* a \beta^*}{\rightarrow}_B r'$ and $\forall (q, r) \in R$: $q \overset{\beta}{\rightarrow}_A q' \implies \exists r'$: $(q', r') \in R \wedge r \overset{\beta^*}{\rightarrow}_B r'$.

A weak bisimulation over A and B is a relation R such that R and R^{-1} are both weak simulations. we say that A and B are *observationally equivalent* and we write $A \sim B$ if for each state of A there is a weakly bisimilar state of B and conversely.

We consider the correspondence between actions of B and B_{SR}^{TT} as follows. To each interaction $a \in \gamma$ of B, we associate

either the multi-party interaction a_{MP}, the binary interaction ok_a or the unary interaction a of B_{SR}^{TT}, depending on whether a is inta-task interaction, external conflicting or external and not conflicting interaction. Other interactions of B_{SR}^{TT} (send/receive interactions) are unobservable and denoted by β.

We proceed as follow to complete the proof of observational equivalence. Among unobservable actions β, we distinguish between β_1 actions, that are interactions between ATC components and TTCC components, and β_2 actions that are interactions between TTCC components and CRP component (namely the reserve and fail). We denote by q_{SR}^{TT} a state of B_{SR}^{TT} and q a state of B. A state of B_{SR}^{TT} from where no β_1 action is possible is called a stable state, in the sense that any β action from this state does not change the state of atomic components.

***Lemma** 2:* From any state q_{SR}^{TT}, there exists a unique stable state $[q_{SR}^{TT}]$ such that $q_{SR}^{TT} \rightarrow_{\gamma^{TT}}^{\beta_1^*} [q_{TT}^{SR}]$

Proof: The state $[q_{SR}^{TT}]$ exists since in AC component no β transitions are possible and since each ATC component $B_{SR\,i}^{TT}$ can either send one or successive offers, or receive a notification. Since two β_1 transitions involving two different components are independent (i.e., modify distinct variables and places), the same final state is reached independently of the order of execution of β_1 actions. Thus $[q_{SR}^{TT}]$ is unique. ■

The above lemma proves the existence of a well-defined stable state for any of the transient states reachable by the B_{SR}^{TT} model. The state $[q_{SR}^{TT}]$ verifies the property $q_{SR}^{TT} \rightarrow_{\gamma^{TT}}^{\beta_1^*} [q_{TT}^{SR}]$ and $[q_{SR}^{TT}] \not\rightarrow_{\gamma^{TT}}^{\beta_1^*}$.

***Lemma** 3:* At a stable state $[q_{SR}^{TT}]$, the B_{SR}^{TT} model verifies the following properties:

- All ATC components are in non busy places $\perp_p^*$ or l.
- All other atomic components are in a non busy place l.
- All TTCC components are in receive places $read$,
- The clock C^g and all variables in ATC components have the same value than their copies in the TTCC component.

Proof: The three first points come from Lemma 1 that guarantees possible execution of a send/receive interaction if its send-port is enabled. Therefore no place $write$ in the TTCC components (respectively $l_\perp$ in atomic components) can be active at $[q_{SR}^{TT}]$, otherwise the answer p_s of TTCC (respectively the offer from $l_\perp$) could occur. Furthermore, since all offers have been sent, none of the TTCC components can be in $wait$ place. In each $TTCC_j$, when $read$ place is active, the last executed transitions are either offers or a fail message reception. The latter does not modify variables in the $TTCC_j$. For each variable x in the $TTCC_j$, the last modifying transitions are offers from the corresponding atomic components B_i, which ensures that each variable in the TTCC component has the same value as the corresponding ATC component. The clock C^g has the same value in the atomic components and the TTCC as it is never reset. ■

***Lemma** 4:* When ATC component $B_{SR\,i}^{TT}$ is in a stable state, we have $n_i > N_i$

Proof: Where in a stable state, the ATC component is either in place $\perp_p^*$ or l following Lemma 3. When ATC component is in place $\perp_p^*$ all offers have been sent, thus the participation numbers in TTCC corresponds to those in components. Initially, for each ATC component B_i, $N_i = 0$ and $n_i = 1$. By letting all components sending offers to all TTCC components, we reach the first stable state where the property holds, since β_1 actions do not modify the N_i variables. The variables N_i in the CRP component are updated upon execution of an ok_a transition, using values provided by the TTCC, that are values from components according to Lemma 3. Thus, in the unstable state reached immediately after an ok_a transition, we have $n_i = N_i$ for each component $B_{SR\,i}^{TT}$ participant in a. Then, the notification transition increments participation numbers in components so that in the next stable state $n_i > N_i$. For components B_i' not participating in a, by induction on the number of $okinteractions$, we have $n_i' > N_i'$. Now when ATC component is in place l, only multiparty interaction is possible with AC components. This interaction involves only AC components, and thus has no effect on N_i, it only increments n_i. If a previous inter-task interaction has been performed so at state l, we have $n_i > N_i$. If no intertask interaction has been performed before reaching place l, the property holds since initially $n_i = 1$ and $N_i = 0$, and since each action from a state l increments n_i. ■

Lemma 4 shows that the participation numbers propagate in a correct manner. In particular, at any stable state the conflict resolution protocol has only previously used values and TTCC components have the freshest values, that are the same as in ATC components. To prove the correctness of the step 1 of our transformation, we exhibit a relation between the states Q of the original model $B = \{B_1, .., B_n\}$ and the states Q_{SR}^{TT} of final model $B_{SR}^{TT} = \{B_{SR\,1}^{TT}, .., B_{SR\,n}^{TT}\}$ and prove that it is an observational equivalence.

We define the relation by assigning to each state $q_{SR}^{TT} \in Q_{SR}^{TT}$ an equivalent state $equ(q_{SR}^{TT}) \in Q$ by:

1) considering the unique stable state $[q_{SR}^{TT}]$ reachable by doing β transitions.
2) considering the control location $l_\perp$ and $\perp_p^*$ in $B_{SR\,i}^{TT}$ as the control location l for B_i, in $equ(q_{SR}^{TT})$. The control location l in $B_{SR\,i}^{TT}$ are considered as the control location l for B_i, in $equ(q_{SR}^{TT})$. Lemma 3 ensures that it is a valid control state for B_i.
3) taking the valuation of variables of $B_{SR\,i}^{TT}$ to a valuation of variables in B_i, and
4) taking the valuation of original clock c_i in B_i as the valuation of $g_{\rho_{c_i}}$.

***Theorem** 1:* $B_{SR}^{TT} \sim B^{TT}$

Proof: We then define the equivalence R by taking:

$$R = \{(q_{SR}^{TT}, q) \in Q_{SR}^{TT} \times Q | q = equ(q_{SR}^{TT})\} \quad (7)$$

The three next assertions prove that R is a weak bisimulation:

1) If $(q_{SR}^{TT}, q) \in R$ and $q_{SR}^{TT} \xrightarrow{\beta} r_{SR}^{TT}$ then $(r_{SR}^{TT}, q) \in R$.
2) If $(q_{SR}^{TT}, q) \in R$ and $q_{SR}^{TT} \xrightarrow{\sigma} r_{SR}^{TT}$ then $\exists r \in Q : q_{SR}^{TT} \xrightarrow{\sigma} r$ and $(r_{SR}^{TT}, r) \in R$.
3) If $(q_{SR}^{TT}, q) \in R$ and $q \xrightarrow{\sigma}_\gamma r$, then $\exists r_{SR}^{TT} \in Q_{SR}^{TT}$: $q_{SR}^{TT} \xrightarrow{\beta^*\sigma}_{\gamma_{SR}^{TT}} r_{SR}^{TT}$ and $(r_{SR}^{TT}, r) \in R$.

The property 1) is a direct consequence of Lemma 2. If $q_{SR}^{TT} \rightarrow_{\gamma_{SR}^{TT}}^{\beta} r_{SR}^{TT}$, then β is either β_1 action, thus by definition $[q_{SR}^{TT}] = [r_{SR}^{TT}]$, or β is β_2 action which does not change the state of the atomic components, thus $[q_{SR}^{TT}] = [r_{SR}^{TT}]$. Thus, we

 ISBN: 978-1-61208-438-1

have $equ(q_{SR}^{TT}) = equ(r_{SR}^{TT})$.

To prove property 2), we assume the action σ in B^{TT} is either an intra-task interaction (multi-party interaction) a, or an inter-task interaction (corresponding to a unary or binary interaction) or a delay step δ.

If a is an intra-task interaction, it is not concerned by the transformation. That is $q = equ(q_{SR}^{TT})$ and $r = equ(r_{SR}^{TT})$.

If a is an inter-task interaction, it corresponds to executing a transition labelled by a unary port a in TTCC component handling a or a transition labelled by ok_a. These transitions are enabled according to valuations of variables and clock C^g in the TTCC handling a. If a is not externally conflicting, by construction of the TTCC, the transition labelled by a has the conjunction of guards g_p sent by the ATCs of different tasks for each $p \in a$. Thus the guard of this transition is $G = G_a \bigwedge(\bigwedge_{p \in a} g_p)$, where $g_p = g_p^X \bigwedge tc_p$ being the conjunction of the boolean guard over variables, and timing constraints. By lemma 3 these values are the same in atomic talking components (ATC), and by extension in $q = equ(q_{SR}^{TT})$. Thus the guard of a evaluates to *True*at $q = equ(q_{SR}^{TT})$. By construction of ATC components, the guard $g_p = g_p^X \bigwedge tc_p$ sent from $l_\perp$ in $B_{SR\,i}^{TT}$ are boolean guards and timing constraints at state l in B_i.

These timing constraints are expressed on clock C^g which could be equivalently expressed on original clocks c involved in the original timing constraints of B_i. The valuation of each clock c involved in the timing constraint of a is computed from the valuation of $g - \rho_c$ where ρ_c is the last clock reset date of c. Therefore, at state q the timing constraint of a expressed on its original clocks are also met.

if a is externally conflicting, the transition labelled ok_a in CRP is possible only if the transition rsv_a executes in the TTCC component. This transition has the same guard and timing constraint of transition labelled by a. Thus, if this transition is possible in the TTCC component, then the guard of a is met at q. Moreover, if transition labelled ok_a is enabled, this means that for each component B_i involved in a, $n_i > N_i$. In particular, for each involved component B_i, the offer corresponding to the number n_i has not been consumed yet. Thus, we conclude that in both cases, we have $q \xrightarrow{a} r$. Finally, executing a in B_{SR}^{TT} triggers the execution of the data transfer function F_a, followed by the computation in ATC upon reception of the response. Thus at $[r_{SR}^{TT}]$, the values in ATC components are the same as in r, which yields $(r_{SR}^{TT}, r) \in R$.

if a is a delay step, it corresponds by letting time progress by δ in either busy locations $l_\perp$ or in places between $wait$ and $read$ of the TTCC components, corresponding to ATC components B_i. Location $l_{\perp\,i}$ has the time progress condition tpc_{l_i} and states between $wait$ and $Read$ in TTCC has each the time progress condition tpc_i sent from the ATC components and corresponding to time progress condition of location l_i. Thus, all these time progress conditions are not false, otherwise the δ delay step would not be allowed.

By Lemma 3, the values involved in time progress condition and sent from $l_\perp$ in $B_{SR\,i}^{TT}$ are the values of time progress condition at state l_i in B_i. These time progress conditions are expressed on clock C^g which could be equivalently expressed on original clocks c involved in the original time progress conditions B_i. The valuation of each clock c involved in the time progress condition of B_i is computed from the valuation of $g - \rho_c$ where ρ_c is the last clock reset date of c. If the time progress condition tpc_i expressed on clock C^g allows the time step δ in TTCC component, thus, δ is also allowed by the time progress condition $tpc_{l\,i}$ expressed on original clocks of B_i. Therefore $q \xrightarrow{\delta} r$. Executing δ has the same effect on clocks in both models, therefore $(r_{SR}^{TT}, r) \in R$.

To prove the property 3), we notice that if σ can be executed in B at state q, then from an equivalent state q_{SR}^{TT}, one can reach the state $[q_{SR}^{TT}]$ (by doing β_1 actions) where the clock C^g and data of ATC components have the same values as those of q (Lemma 3). As previously, we distinguish the cases where σ is an interaction a or a delay step δ.

If σ is an intra-task interaction a, then it is not concerned by the transformation and remains intact in the obtained model. Thus we have straightforwardly $(q_{SR}^{TT}, q) \in R$ and $(r_{SR}^{TT}, r) \in R$.

If σ is an inter-task interaction a from q, then the timing constraint and guard of a are *True*. From q_{SR}^{TT} we reach $[q_{SR}^{TT}]$ by doing β_1 actions. Then, we execute all possible $fail$ interactions (that are β_2 actions), to reach $[q'^{TT}_{SR}]$. At this state, if a is not conflicting, the interaction a is enabled, else the sequence $rsv_a\ ok_a$ can be executed since lemma 4 ensures that guard of ok_a is *True*. In both cases, the interaction corresponding to a brings the system in state r_{SR}^{TT}. From this state, the response corresponding to ports of a are enabled, and the next stable state $[r_{SR}^{TT}]$ is equivalent to r. Thus we have $q_{SR}^{TT} \xrightarrow{\beta_1^*} [q_{SR}^{TT}] \xrightarrow[fail]{\beta_1^*} [q'^{TT}_{SR}] \xrightarrow{a} r_{SR}^{TT} \xrightarrow{\beta_1^*} [r_{SR}^{TT}]$ and $q_{SR}^{TT} \xrightarrow{\beta_1^*} [q_{SR}^{TT}] \xrightarrow[fail]{\beta_1^*} [q'^{TT}_{SR}] \xrightarrow{rsv_a, ok_a} r_{SR}^{TT} \xrightarrow{\beta_1^*} [r_{SR}^{TT}]$. Thus $(r_{SR}^{TT}, r) \in R$.

If σ is a delay step δ then from state $[q_{SR}^{TT}]$, time can progress also by δ to reach state r'^{TT}_{SR}. At this state, place $read$ of TTCC is active and having the tpc_i sent by $[B_{SR\,i}^{TT}]$ which is the same as the time progress condition of B_i at q. From reached state r'^{TT}_{SR} we execute all possible $fail$ interactions (that are β_2 actions), to reach r_{SR}^{TT}.The time progress conditions are equal to *False*in $receive$ place in the CRP component. That is, even if some β_2 actions is possible from state r'^{TT}_{SR}, no delay is allowed at this state. Thus we have, $q_{SR}^{TT} \xrightarrow{\beta_1^*} [q_{SR}^{TT}] \xrightarrow{\delta} r'^{TT}_{SR} \xrightarrow{\beta_2} r_{SR}^{TT}$ with $(r_{SR}^{TT}, r) \in R$. ■

C. Trace inclusion between B_{SR}^{TT} and B^{TT}

We denote by B_{SR}^{TT} the initial model and by B^{TT} the resulting model of the step 2 of the transformation.

The observational equivalence cannot be proven between B_{SR}^{TT} and the resulting B^{TT}. In B^{TT} model, conflicts between internal and external interactions are resolved by adding local priority rules in the task. Thus when interactions a_I and a_E (respectively internal and external interactions) are possible, always the internal one will be executed. This restriction implies that the set of traces of the B^{TT} model is a subset of traces of B_{SR}^{TT} model.

A User-App Interaction Reference Model for Mobility Requirements Analysis

Xiaozhou Li*, Zheying Zhang†
School of Information Sciences, University of Tampere
Email: *li.xiaozhou.x@student.uta.fi, †zheying.zhang@uta.fi

***Abstract*—Contemporary mobile applications (apps) value mobility as a key characteristic that allows users to access services or features ubiquitously. In order to achieve decent mobility, apps shall provide features that are suitable to use under a wide range of contexts. In this paper, we analyze the situational contexts, towards which the mobile apps shall comply with in terms of mobility. By analyzing the contexts and the ways of interaction between users and apps, we propose and illustrate a mobile requirements analysis process model to identify the conflicts between users' ideal ways of interaction and the way the feature is designed to provide. The identified conflicts help to elicit requirements for the enhancement of the apps' mobility.**

Keywords–Mobility; Mobile application; Requirements; Context; Situational Context; Interaction;

I. INTRODUCTION

The emergence of iOS and Android OS has been changing the mobile industry and people's daily lives, and been providing new trends in the academic research [1]. Changes in distribution process and mobile software market mechanism lead to better customer accessability towards mobile apps and inevitable competition [2]. The ranking mechanism also intensifies the competition, demanding mobile apps satisfying users' diversified demands, which shall be reached in varying situations, compared to desktop software. It thus requires companies to take into account the capability of the mobile applications to provide satisfactory user experience regardless the changing environment [3].

Mobility is one of the most significant and unique features for mobile apps, refering to the ability to access services ubiquitously through wireless networks and various mobile devices [4][5]. The vision of mobility is to be able to work "anytime, anywhere" [6]. However, with limited support of systems towards mobility, the capability of being comfortably used at "anytime, anywhere" of mobile apps is seldom achieved. Thus, achieving mobility shall result in the enhanced competitiveness of mobile apps in terms of user satisfaction.

Contexts, which refers to the information that characterizes the situation of an entity, has a great impact on usability and user experience of mobile apps [7]–[10]. Mobility, as a key aspect of usability of mobile apps, is also affected largely by their context [11], which also influences the way, by which a user interacts with an mobile app [12]. By specifying the ways, in which user and app interact, we analyze their relation towards different contexts. The elicited mobility requirements shall thus reflect suitable interaction ways between users and apps in differen contexts.

Many studies analyze the phenomena of use situations concerning mobile commerce [4][13] and other types of mobile apps [14]. But studies on mobility requirements analysis are very limited. A goal-oriented framework for modeling and analyzing requirements for varying contexts was proposed based on the goal model to reason variants [15]. But it fails to address how to identify the varying contexts and derive requirements regarding the way of interaction. Several other studies [16][17] address challenges and methods of requirements analysis for mobile systems and pervasive services, but lack a concrete proposal on taking varying contexts into account for requirements analysis.

In this study, we focus on the analysis of context of use of mobile apps and the possible ways of interaction between users and an app, and further study the way of analyzing mobility requirements. The paper tries to tackle the following questions.

- RQ1 *What are the contexts that affect the interaction between user and mobile apps?*
- RQ2 *What are the ways in which a user interacts with mobile apps, and what are their relations with different contexts?*
- RQ3 *How to take into account contexts and ways of interaction when analyzing apps mobility requirements?*

The purpose of this research is to enhance the mobility of mobile apps by taking into account the varying contexts in requirement analysis. To answer RQ1, we summarize the definition of mobility and main perspectives of mobile app contexts by reviewing the literature on the concept of mobility and context in Section 2. We also tackle RQ2 by analyzing the relation between the way users interact with mobile apps and different contexts via user-app interaction reference model in Section 3 and 4. In Section 5, we propose our approach to analyzing mobility requirements with a case study for further illustrating and discussing in Section 6, which altogether answers RQ3. Section 7 concludes with implications for future research.

II. MOBILE APPS AND THEIR MOBILITY REQUIREMENTS

Mobile devices and applications has enabled new freedom and flexibility on the way people communicate, work, and entertain by providing services beyond the constraints of fixed locations and devices. Compared to the old style of using manufacturer provided mobile software, contemporary mobile apps have lower distribution costs and can be more easily accessed by customers via the change in distribution process and mobile software market mechanism [2]. The mechanism stimulates the development of mobile application markets and results in fierce competition with even software on personal computers challenged. Prior to the investigation of the circumstances, under which users would prefer to mobile applications rather than desktop software, the unique mobility characteristics of mobile applications differentiated from those of desktop software shall be understood.

A. Mobility

Mobility was understood as the human's independency from geographic constraints or the ability and/or quality to ensure the given entity can move or be moved [18]. Mobility primarily facilitates a mobile device to operate properly when

 ISBN: 978-1-61208-438-1

its location changes. This provides a generic view of how mobility is supposed to be acquired by users when they use mobile apps, and forms a general goal in mobile app development. According to [5], the key feature of mobile technology is the capability of using services on the move, with wireless network and various devices, which provides the literal meaning of mobility. Other similar terms, such as nomadicity, which indicates a system's capability of providing services to the nomad as he moves from place to place in a transparent and convenient form [6], also reflexes the concept of mobility.

Consequently, mobility is an attribute of both human beings and the computational devices they interact [11]. The mobility of mobile devices refers to the ability to access services ubiquitously, or "anytime, anywhere" through wireless networks and various mobile devices [4][6] and for mobile apps as well. As a critical feature of mobile apps usability, mobility has considerable impact on the interaction between users and mobile devices and apps [19]. Thus, compared to mobility of human beings, mobile app mobility is seen as the usefulness and ease of use provided by the app towards user satisfaction in "anytime, anywhere".

There are three types of mobility in terms of modality [20], i.e., travelling, visiting and wondering. The mobility towards the usability of mobile apps is hence seen in these three perspectives as well, that is, to provide services when users are traveling, visiting, and wondering. Similar categorization is also given by [21], which describes the motion of mobile app users into none motion, constant motion and varying motion. Besides the categorization of mobility regarding spatial movement, time and context changes also contribute to the mobility attribute provided by mobile apps [18]. In this study, we see all the factors that influence the mobility of the mobile app from outside the app itself as its context.

From the context of use perspective, mobility implies that the app shall provide context-aware features and/or services. Following the definition of the concept of context, i.e., the information that can be used to characterize the situation of an entity [7][8], where an entity can be a place, person, physical or computational object, we define context-aware features as the use of context to provide task-relevant information and/or features, which a user feels easy to use. The situation can be characterized in perspectives, such as location, surrounding changing objects, and people, and can define where you are, who you are with, and what resources are nearby [22][23]. These perspectives can be further refined into users, tasks, equipment (i.e., hardware, software and materials), location, physical environment, temporal context, social environment, technical and information context, etc. and have been intensively addressed and adapted in surveys and research on the context in mobile computing and the impact on the overall design of the product [8][10][17][24]–[28] .

B. Mobility Requirements Analysis

We consider mobility as an intrinsic attribute of mobile applications. It refers to the capability of providing receptive and pleasant services acquired by users in spite of the changes in environments. Such an attribute can be refined into different types of requirements contributing to users' satisfaction. The requirements include functional requirements complementing the main features of an application and supporting users to fulfill their goals, interface requirements facilitating the interaction between users and the application, as well as constraints on the application.

In addition to analyzing the core features of an application, mobility requirements analysis shall emphasize ease of use in the dynamic environment of use, and focus on analyzing the diversity of context of use and ways of interaction between users and the app. Accordingly, we adapt the generic requirements syntax of Mavin et al.'s EARS model [29] for mobility requirements, emphasizing the context and the interaction with users, as shown below.

In **<situational contexts>,**<optional preconditions> <optional trigger> **the** <mobile app name> **shall** <app response> **in <ways of interaction>.**

The syntax marked in grey is what was specified in the EARS model [29]. It can be further specialized into different types of requirements following temporal logic defined between the precondition, the trigger, the app response, etc. [29]. In addition, the components marked in black are situational contexts and ways of interaction, which highlights the mobility attributes a mobile app shall reflect and the requirements analysis shall take into account. The situational context encompasses a wide range of elements, such as the location and surroundings, the social context, the user's movement, the temporal context, etc. Combining value of these elements forms a variety of scenarios of using a mobile app. Changes of the scenarios continuously reframe a user's interaction with a mobile app. Obviously not all scenarios are desired and friendly. The requirements analyst shall be aware of the suitable ways of interaction is adopted towards typical scenarios, which secures users' satisfaction and receptiveness largely.

III. A User-App Interaction Reference Model

Interaction between a user and a mobile app occurs after the user opens the app and before he or she closes it [30]. We call it a user-app interaction. According to the definition of context given by [7][8], the context of a user-app interaction is referred to as the information to characterize the situation of the two entities, i.e., the user and the app. The context is further depicted in Figure 1.

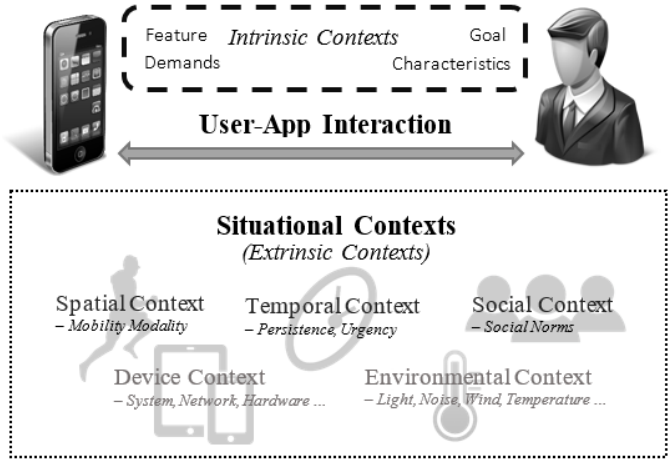


Figure 1. User-App Interaction Reference Model.

The context shall ideally contain all possible situations that affect the user-app interaction. Similar to the contexts categories described in previous studies [17][22]–[24], the ideal context shall contain multiple perspectives including user profile, operating system, hardware system and network, physical context, temporal context, task context, and social context. As shown in Figure 1, we divide the wide range of context into intrinsic and extrinsic ones. The intrinsic context refers to the inner attributes of entities that influence, or occasionally determine the occurrence of a user-app interaction. For example, the features provided by mobile apps and demands for operating them intrinsically determine their usefulness when users' goals and their characteristics determine whether to use. The extrinsic context refers to the external factors that influence a user's decision of his or her engagement in a user-app interaction. The extrinsic contexts, defined also as situational contexts, include device context, environmental context, spatial context, temporal context and social context. The device context (e.g. system, network, hardware, etc.) and the environmental context (e.g. light, noise, wind, temperature, etc.) have been often studied in requirements engineering for self-adaptive software systems [31]. However, the other extrinsic contexts, such as spatial, temporal, and social contexts are rarely discussed in requirements analysis process. Hereby, we focus on these situational contexts to investigate their relations with the mobility attribute of a mobile app.

A. Intrinsic Contexts

Many researchers have addressed and verified that a user's demographic properties, such as age, gender, education, income, etc. are relevant factors affecting a user's attitudes and preference to the use of an app [17][28][32]–[34]. Besides the demographic properties, individual users with various interests and attitudes toward the mobile value form other important perspectives influencing a user's engagement in an app activity. Users' adoption to mobile services has been analyzed from the perspectives of a user's characteristics, major value, attitude, and major interests. [28]. Concerning a user-app interaction, the user's goals refer to the objectives of the user at the critical moment when the interaction occurs. For an instance, the goal of a student working on an exam is to pass the exam. It is not only related to mobile applications but also has a wider range than the term described in goal-oriented requirements engineering [35]. Thus the characteristics and goals of the user form the intrinsic and determinant context to the initiation of a user-app interaction [36].

On the other hand, a user-app interaction occurs when a user determines to perform a task associated with a mobile app with a particular purpose, i.e., fulfilling a user's goal. The features of a mobile app contain capabilities that enables users to fulfill their goals by accomplishing the tasks. The user is prone to intrinsically start an interaction with the app when the provided features comply with his or her intention to achieve his or her goal. The demands are defined as the workload that a user is obliged to engage in order to accomplish the task as the demands, which contain six subscales, i.e., mental demand, physical demand, temporal demand, frustration, effort and performance [37]. They define the subjective experience of users on using the app and are affected by the intrinsic contexts of a user-app interaction as well.

B. Situational Context Model

The situational context refers to the extrinsic properties of the user and the app that impact the initiation of a user-app interaction. As mentioned above, we only focus on the temporal, spatial and social perspectives to discuss the situational context in this study.

1) Temporal Context: Temporality, as one of the dimensions of the mobility concept originally [18], has been influenced by the mobile technology inherently in terms of human interaction. The multiple perspectives of temporality, such as structural and interpretive, monochronicity and polychronicity and so on have been studied previously [18][38]. Compared to the previous frameworks, we argue that the temporality in terms of a user-app interaction is determined by the user's sense of time and the persistence of the app to accomplish one operation session, and define temporal context in this paper as the sense of external time pressure of the user caused by the confliction or the accordance of user's goal and app's demands. Thus, two values of intensive and allocative are used to describe temporal context. Intensive refers to the situation when the user is in urgent need of achieving his or her goal and has limited spare time of interacting with the app (e.g. the user is busy in working on assignments with approaching deadline). Allocative, on the other hand, indicates that the user has no urgent goal to achieve and is temporally available (e.g. the user is staying at home idle).

2) Spatial Context: The spatial perspective of mobility indicates the geographic movement of the user when engaged in the interaction with the mobile app [18]. In this study, the spatial context refers to the current movement of the user further indicating the physical availability for the app usage. we adopt the mobile modality types given by [20] categorizing the spatial context, including visiting, traveling and wandering. Visiting context indicate that the user is in a physically stationary status(e.g. sitting in a meeting). Traveling context refers to the situation when the user is in a transportation tool (e.g. a car or train). Wandering, on the other hand, refers to the situation when the user is physically moving from place to place (e.g. walking or running). However, the categorization given by [20] did not specify the difference between driving a transportation tool or sitting in one in terms of traveling perspective. In addition, exercise related scenarios of walking or running is not taken into account either. In this study, these distinctions shall be reflected by the combination with other contexts.

3) Social Context: The social context is interpreted by [8][10][17][24]–[28][39] as the influence of other persons' presence and the interpersonal interaction between the user and others. In this study, we interpret the social context of a user-app interaction as the social norms that constrain user from or encourage user into the interaction [40], which contains similar meaning towards the functional place concept in [39]. We define the scale of social context from constraining to encouraging the use of apps based on the social norms. For example, a conference presentation is socially constraining when idleness at home is socially encouraging.

According to the three perspectives of situational context mentioned above, values are assigned to each perspective, combining which leads to a unique context scenario description (shown in Table I).

Ideally, based on the given situational context model, the situational context of user can be described by the combination

 ISBN: 978-1-61208-438-1

TABLE I. VALUES OF SITUATIONAL CONTEXT PERSPECTIVES.

Perspective	*Value*
Temporal	**I**ntensive, **A**llocative
Spatial	**V**isiting, **T**raveling, **W**ondering
Social	**C**onstraining, **E**ncouraging

of the three perspectives. The 12 situational contexts include *Intensive-Visiting-Constraining (IVC), Allocative-Visiting-Constraining (AVC), Intensive-Visiting-Encourage (IVE), Allocative-Visiting-Encourage (AVE), Intensive-Traveling-Constraining (ITC), Allocative-Traveling-Constraining (ATC), Intensive-Traveling-Encouraging (ITE), Allocative-Traveling-Encouraging (ATE), Intensive-Wondering-Constraining (IWC), Allocative-Wondering-Constraining (AWC), Intensive-Wondering-Encouraging (IWE), and Allocative-Wondering-Encouraging (AWE)*. For each combination of values from different perspectives, we provide a typical situational context scenario, shown in Table II.

TABLE II. TYPICAL SCENARIOS FOR EACH SITUATIONAL CONTEXT.

Situational Context	*Typical Scenario*
IVC	In a conference giving presentation
AVC	In a lecture listening
IVE	In a cafe working on assignments with close deadline
AVE	At home idle
ITC	In a car driving with time limit
ATC	In a car driving and sight seeing
ITE	In a train when it is about to arrive at the destination
ATE	In a train idle
IWC	Running in a race
AWC	Wondering in a cocktail party as a host
IWE	Running to catch a bus
AWE	Walking in a park relaxing

In practice, the scenarios that used for describing situational contexts might vary based on the collective understanding of the contexts from the team. For example, the scenario "In a conference giving presentation" and "in a contract signing meeting negotiating" can both be used describing the situational context of IVC.

IV. WAYS OF USER-APP INTERACTIONS

The concept of mobility is not only just a matter of people traveling, but also the interaction people perform, that is, the way in which they interact with each other [18]. The mobility is thus reflected in the way in which users interact with the apps. It occurs when an app sends out a notification to the user who responds it and ends when the user finishes using the app and closes it. However, users in different situational contexts, who have different goals and characteristics, will expect to interact with different features of the app differently but comfortably. In order to find the match between the designed and expected ways of interaction, we adapt the dimensions of interaction modality [41][42] discussing the situational characteristics of mobile apps including their obtrusiveness and persistence.

An obtrusive interaction imposes obligation to notice or react [18], which indicates that the interaction is evoked by notifying the user to start it without the user's internal motivation to do so. For example, an obtrusive interaction is initiated when the user stops original reading activity and responds to the new message notification from WeChat. On the contrary, an unobtrusive interaction is initiated with the user's internal motivation. For example, the user encounters an unfamiliar term while reading and decides to look it up in Eudic without receiving notification. On the other hand, the persistence dimension specifies the duration of an interaction, which is largely depending on the time length a user spends on completing an interaction task. An ephemeral interaction requires a short time to achieve user's goal (e.g. replying a message, looking up a word). A persistent interaction oppositely takes a long period to accomplish (e.g. playing Subway Surfers, listening to Spotify).

With the two dimensions combined, a user-app interaction can thus be described as *obtrusive-persistent (OP), unobtrusive-persistent (UP), obtrusive-ephemeral (OE), or unobtrusive-ephemeral (UE)*. By analyzing the relation between different types of user-app interactions and the way they fit in the process of the way of the user's original activity, we conclude four ways of interaction, including *intermittent*, *interrupting*, *accompanying*, and *ignoring*.

An intermittent way of interaction refers to the interlaced engagement in both the user's original task and the user's interaction towards the mobile app, with the whole process of several short interactions, which are neither consistent nor interfering the proceeding of the original task. For example, when watching TV, the user starts the interaction with WeChat. Within the whole process, the user inconsistently responds messages but his or her task of watching TV remains proceeding. An intermittent way of interaction often consists of a number of ephemeral interactions, which are also mostly obtrusive.

An interrupting way of interaction requires the user to convert full concentration on the interaction and cease the original activity. For example, to start playing Subway Surfers, the user has to stop the original task, such as reading books or watching TV. It can be interpreted as the original task is interrupted by this user-app interaction. An interrupting way of interaction is mostly persistent.

An accompanying way of interaction refers to the paralleling engagement in both user's original task and the user-app interaction tasks. Comparing to the interrupting or the intermittent way of interaction, the accompanying one will not attract the user's full attention, as the user does not stop the continuous progress of the original task. For example, when running on a treadmill, the user starts watching films from Netflix. The activity of running is, instead of interrupted, paralleling with the user-app interaction. The interaction can be ephemeral or persistent, depending on the amount of engagement an app requires from the user.

An ignoring way of interaction indicates that the interaction with the mobile app is ignored by the user in order to maintain the proceeding of his or her original task. For example, when taking an examination at school, the user will ignore any types of interaction with the mobile apps.

The relation between different types of user-app interactions and the according ways of interaction is summarized in Figure 2. In Figure 2, the narrow arrows underneath represent the user's original task, and the thick ones represent the user-app interactions. Lighter gray arrows are the interactions ignored and not executed. In addition, the length of the arrows indicates the timeline of proceeding with the task or interaction.

By analyzing the different ways of user-app interactions, we are enabled to analyze the expected way of interactions towards each mobile app feature. And towards mobility, expected ways of interaction shall comply with the previously

 ISBN: 978-1-61208-438-1

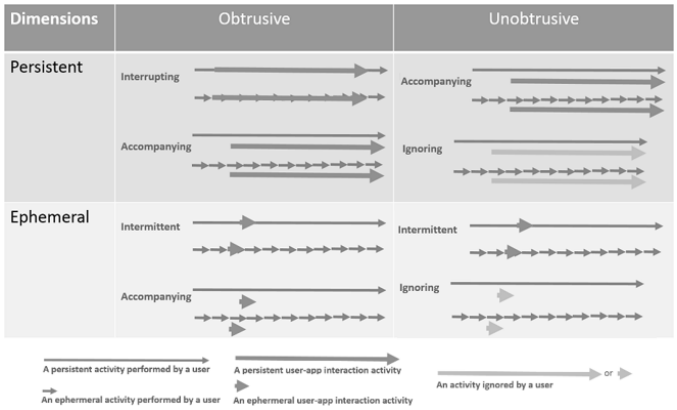


Figure 2. Ways of User-App Interactions

defined situational contexts. Combining this analysis, we shall be able to detect how an app feature is expected to perform in different situational contexts. For requirements analysts, each feature and the refined requirements could be analyzed and mapped into the situational contexts assigned with expected ways of user-app interactions. For example, Table III shows the expected ideal ways of interaction towards given situational context .

TABLE III. IDEAL WAY OF INTERACTION FOR EACH SITUATIONAL CONTEXT.

S.C.	*Typical Scenario*	*Ideal Ways*
IVC	In a conference giving presentation	Accompanying
AVC	In a lecture listening	Intermittent
IVE	In a caf working on assignments with close deadline	Intermittent, Accompanying
AVE	At home idle	Interrupting Intermittent Accompanying
ITC	In a car driving with time limit	Accompanying
ATC	In a car driving and sight seeing	Accompanying, Intermittent
ITE	In a train when it is about to arrive at the destination	Accompanying, Intermittent
ATE	In a train idle	Interrupting, Intermittent Accompanying
IWC	Running in a race	Accompanying
AWC	Wondering in a cocktail party as a host	Intermittent
IWE	Running to catch a bus	Accompanying
AWE	Walking in a park relaxing	Accompanying, Intermittent

Table III indicates that in a specific situational context one mobile app feature shall enable users to interact comfortably in the ideal ways of interaction. Taking the situational context scenario of IVC as an example, the ideal way is the accompanying way of interaction. Thus, a mobile app feature which offers such a way of interaction is more likely to be used in this circumstance. For example, the slide presentation feature of Prezi can provide accompanying way of interaction in the IVC context scenario of "In a conference giving presentation". The typical scenario of situational contexts can be also different. For example, IVC context can also be represented in the scenario of "In a university exam with time limit" or "In a chess competition with time limit for each move". When a certain feature fails to initiate the ideal ways of interaction, it shall be adjusted at requirement specification level towards the ideal ways. Therefore, a process of identifying such features and specifying the according strategy of mobility enhancing adjustment is required.

V. MOBILITY REQUIREMENTS ANALYSIS PROCESS

The mobility requirements analysis process contains a sequence of pre-defined steps, by following which requirements analysts can specify existing user requirements towards enhanced mobility. The aim of mobility requirements analysis is to provide specified requirements that enable users to use the given features in a satisfied way in the possible situational contexts. As defined previously, a user's satisfaction for a specific feature is achieved by using this feature in different situational contexts via ideal ways of interactions. Thus the proposed mobility requirements analysis process is to refine existing app features by taking into account the given situational contexts and the according ways of interactions. The process of mobility requirements analysis is described as Figure 3.

```
Input: Requirements Document (RD)
Output: Mobility enhanced Requirements Document (MERD)

Identify Primary Situational Contexts
for each feature in RD, do
        Specify Interaction
        for each specified interaction, do
                for each primary situational context, do
                        if way of interaction ∉ the ideal way of interaction,
                        then change the requirements towards primary situational context;
                             document the requirements and changes into MERD;
return MERD
```

Figure 3. Mobility Requirements Analysis Process.

The analysis process consists of four key steps, as explained below.

Step 1. Identify the Primary Situational Contexts

Mobile apps are meant to satisfy users' needs in all possible situational contexts in an ideal way. However, mobile apps have distinct visions and features, and cannot comply with every situational context to meet users' needs. It thus requires the requirements analysts to identify the primary situational contexts by prioritization. The outcome of this activity is a list of prioritized situational contexts, or a number of primary situational contexts.

Step 2. Specify the Expected Way of Interaction for Each Feature

For each feature of the mobile app, requirements analysts shall be able to specify an expected way of interaction, which is expected by users. It means that users will use this feature most comfortably via that way of interaction. The outcome of this activity is a list of features together with expected ways of interaction respectively.

Step 3. Compare the Previous Two Outcomes and Identify the Conflicts

By comparing the outcomes of the previous steps with Table III, we can find the features, of which the expected way of interaction, conflicts with the ideal ones of the primary situational contexts. These conflicts shall be adjusted in the next step to enhance the app's mobility attribute.

Step 4. Adjust Conflicting Feature towards Mobility Requirements

We diminish the conflicts by changing the requirements related to the feature or adding new ones.

VI. CASE STUDY

By following the steps of mobility requirements analysis process, we are able to identify the features of a mobile app that may contain conflicts against the ideal ways of

 ISBN: 978-1-61208-438-1

interaction in specified primary situational contexts, and also to analyze and adjust the according features towards eliminating the conflicts, hence enhancing their mobility. In this section, we apply our proposed approach to analyzing three mobile apps, WeChat[43], Gmail[44], and AlienBlue[45]. WeChat is a messaging and calling app. It allows users to communicate with friends for free text (SMS/MMS), voice & video calls, moments, photo sharing, and games. Gmail (IOS) is the official mobile app for iPhone and iPad. It supports real-time notifications of new mails, multiple accounts, and mail search across the entire inbox. AlienBlue is the official app for Reddit, an online bulletin system. It enables users to browse threads from Reddit, post new threads and reply on others' threads with other features, such as liking or disliking, subscribing, image uploading, and so on.

The three mobile apps share the essential feature of user communication but contain differences in details. For example, WeChat enables users to receive, send, and share multimedia messages instantly. Gmail contains no voice messaging feature and takes longer time on individual operation session, such as browsing and replying emails. On the other hand, the communication between users on AlienBlue is fulfilled by posting, reading and replying threads in Reddit. Thus, in the study, we focus on the communication feature of the three apps to analyze their mobility attributes and requirements.

Step 1. Identify the Primary Situational Context

Firstly, the primary situational contexts shall be identified amongst the previously defined 12 situational contexts, as well as the scope of the analysis. Instead of prioritizing the 12 situational contexts precisely, for these cases, we categorize situational contexts into three prioritization level, including, *primary*, *secondary* and *ignorable*. Primary situational contexts indicates that most users tend to use this feature in these situational contexts. Secondary contexts are those situational contexts in which the user has the equal possibility of using the app or not. And ignorable contexts are those in which users nearly never use the feature. In terms of the "user communication" feature of the three cases, the according categorization of situational context is shown as Table IV.

TABLE IV. PRIMARY SITUATIONAL CONTEXTS.

	WeChat	*Gmail*	*AlienBlue*
Primary	AVE, ATE, AWE	AVE, ATE	AVE, ATE
Secondary	AVC, IVE, ITE, AWC	ITE, AWE	AWE
Ignorable	IVC, ITC, ATC, IWC, IWE	IVC, AVC, IVE, ITC, ATC, AWC, IWC, IWE	IVC, AVC, IVE, ITC, ATC, ITE, IWC, AWC, IWE,

Taking WeChat as an example, it enables users instant communication. Thus, the actions of reading and replying messages is to a large extent encouraged in the situation without social constraints (e.g. driving for safety reason) or time limit towards other objectives (e.g. deadlines). The social encourging and time allocative contexts are also the primary contexts for the other two apps. But different from them, instant communication feature of WeChat is also encouraged in 'wondering' contexts. Besides, even with certain social constraints and time limits, users tend to use WeChat more than the other two, which is why more secondary situational contexts are identified for WeChat.

Step 2. Specify the Expected Ways of Interaction

The expected way of interaction for the feature shall be determined by the way in which most of the users use the feature, which can be identified and analyzed by using different requirements elicitation techniques, or asserted by requirements analysts based on the use pattern of other similar products. For example, the expected way of interaction for WeChat communication is *intermittent*, as users only allocate short time for instant communication without original activity fully interrupted. Comparatively, Gmail and AlienBlue require more concentration and time from users for reading and replying emails, which results in an *interrupting* way of interaction.

Step 3. Compare the Previous Two Outcomes and Identify the Conflicts

Comparing the pre-defined ideal ways of interaction for the primary situational contexts and the expected way of interaction for the feature, we find no conflicts for all apps in their primary situational contexts (shown in Table V). When no conflicts are found for all primary situational contexts, we can indicate that this existing feature provides adequate mobility support.

TABLE V. COMPARISON OF IDEAL AND EXPECTED WAYS OF INTERACTION

Primary	*Ideal*	*WeChat*	*Gmail*	*AlienBlue*
AVE	Interrupting, Intermittent, Accompanying	Intermittent	Interrupting	Interrupting
ATE	Interrupting, Intermittent, Accompanying	Intermittent	Interrupting	Interrupting
AWE	Intermittent, Accompanying	Intermittent	**Interrupting**	**Interrupting**

However, conflicts are found in the secondary context of AWE for Gmail and AlienBlue. Compared with the primary contexts, we find that users in "wondering" context are more likely to start interaction with WeChat rather than Gmail and AlienBlue based on their expected ways of interaction. Thus, to enhance the mobility of them, conflicts for this secondary situational context shall be addressed with additional mobility requirements as in practise the secondary situational contexts might be also of high priorities.

Step 4. Adjust Conflicting Feature towards Mobility Requirements

Once the conflicts were detected, the according feature or function shall be adjusted in order to improve the overall mobility of the app. According to the conflicting situational context (i.e., AWE), the mobility requirement to adjust is as follows.

In a situational context of "Allocative-Wandering-Encouraging", the Gmail/AlienBlue app shall provide user text-based communication functionality in the intermittent way of interaction.

The mobility requirements provide the goal for requirements analysts indicating which specific situational contexts and by which ways of interaction the target feature shall be adjusted. The adjustment can be applied by adding or editing the existing requirements related to this function. Taking AlienBlue as an example, part of functions related to text-based communication in thread discussion is summarized in Table VI.

When adjusting the functions, we shall take into account the conflicting situational context. The perspective that plays a critical part of the conflict is firstly focused. For example, concerning the specific situational context of "Allocative-

 ISBN: 978-1-61208-438-1

TABLE VI. ALIENBLUE'S FUNCTIONS

App Name	*Functions*
AlienBlue	AFR1.The app allows the user to view through the whole thread; AFR2.The app allows the user to reply on a specific comment; AFR3.The app allows the user to send replies with images and emojis; AFR4.The app allows the user to like or dislike other comments; AFR5.The app allows the user to receive comments notifications;

Wandering-Encouraging", social encouraging and time allocative contexts do not hinder user's interaction with the thread receiving and replying feature of AlienBlue. Thus, within the range of this very feature, we change the existing function with more specified function variations. As follows, based on the given functions, we provide examples on how to change the existing function or add new functions that comply with the "Wandering" situational context.

Taken as examples, AFR1 and 2 are two of the essential function of the AlienBlue app, which must not be removed. However, as a persistent and non-obtrusive functions, based on Figure 2, these functions are prone to be ignored in most situational contexts, especially for the "Wandering" context. One way to change them is to shorten the operating session.

Thus, the AFR1 and AFR2 can be changed into:

AFR1.1 *The app shall allow the user to view exclusively his or her own comments and the ones he or she comments on with unrelated comments folded;*

AFR1.2 *The app shall allow the user to view unrelated comments by unfolding them;*

AFR1.3 *The app shall allow the user to quickly control the display of the thread interface by hand gestures;*

AFR1.4 *The app shall allow the user to view comments concerning him or her on the lock screen;*

AFR2.1 *The app shall allow the user to reply with predefined quick responses;*

AFR2.2 *The app shall allow the user to save unfinished comments automatically and to continue composing;*

AFR2.3 *The app shall allow the user to respond to the received comments on the lock screen;*

Compared to the originial requirements, the specified requirements largely reduced the browsing time by directly enabling the user to focus on the relevant comments. Meanwhile, the specified requirements also enhance the obtrusiveness of the notification, which allows the user to better respond to the notification. In this way, based on the existing functions, these functions are adjusted in order to eliminate the conflicts between ideal way of interaction in a certain primary situational context and the expected way of interaction of this feature. By repetitively doing so with all the features of the mobile app, the mobility of the target mobile app is supported as users are enabled to interaction with the features in the expected way of interaction.

In this case study, we adopt existing mobile apps as examples to demonstrate how the mobility requirements analysis process can be applied. It is easy to identify the features that require mobility enhancement and the corresponding change proposal for well-known apps. In practise, it is hard to predict and fully specify all situational contexts and assure that the target app attract users to use in their expected way. The proposed approach and process provides a way of analyzing situational contexts, in which the app is put to use and eliciting requirements that enhance its mobility. This study contributes in filling the gap in the studies on applying the understanding of context into mobile app requirements analysis. Furthermore, the user-app interaction reference model and the situational context analysis provides an extensible framework of studying the context of a user-app interaction where more perspectives can be added to enrich the scenario set of situational contexts. This approach also enables developers to choose the suitable set of context scenarios and prioritization, as well the ideal ways of interactions, based on the vision and scope of their target mobile apps.

VII. CONCLUSION

In this paper, we explore the concept of mobility as the characteristic of mobile apps, which satisfies users' need to use them under changing contexts. By analyzing the perspectives of mobility, we define situational contexts as the key extrinsic factors that influence users' satisfaction in user-app interactions. Compared to the other context factors, such as device context and environmental context, the situational contexts are more tangible towards the understanding of how users and apps interact, and also the factors shall be taken into account when mobile development team aims to enhance the mobility of their mobile products.

Furthermore, based on the specification of typical situational context scenarios, we further analyze connection between these situational contexts and the ideal ways of user-app interactions. Hence, seeking the conflicts between ideal ways of interaction and the current ones is the method to detect the key mobility-lacking features of a mobile app. On the basis of the analysis, we propose the mobility requirements analysis process, which helps to adjust features and the according requirements towards the ideal ways of interaction. Accordingly, the overall mobility of the mobile app improves.

The future work of this study will focus on the other extrinsic contexts and their influences on user-app interactions, which shall be utilized as the replenishment for the existing reference model. The user characteristics and goals, as well as their connection towards the mobile application feature and demands, shall also be reviewed and analyzed in the mobile app domain. In addition, the connection between the improvement of mobility and user satisfaction to mobile apps shall be also studied as the validation of our mobility requirements analysis method in our future studies.

REFERENCES

[1] D. Gavalas and D. Economou, "Development platforms for mobile applications: Status and trends," Software, IEEE, vol. 28, no. 1, 2011, pp. 77–86.

[2] A. Holzer and J. Ondrus, "Mobile application market: A developers perspective," Telematics and informatics, vol. 28, no. 1, 2011, pp. 22–31.

[3] C. Ryan and A. Gonsalves, "The effect of context and application type on mobile usability: an empirical study," in Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38. Australian Computer Society, Inc., 2005, pp. 115–124.

[4] N. Mallat, M. Rossi, V. K. Tuunainen, and A. Öörni, "The impact of use situation and mobility on the acceptance of mobile ticketing services," in System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on, vol. 2. IEEE, 2006, pp. 42b–42b.

[5] C. Coursaris and K. Hassanein, "Understanding m-commerce: a consumer-centric model," Quarterly journal of electronic commerce, vol. 3, 2002, pp. 247–272.

[6] L. Kleinrock, "Nomadicity: anytime, anywhere in a disconnected world," Mobile networks and applications, vol. 1, no. 4, 1996, pp. 351–357.

 ISBN: 978-1-61208-438-1

[7] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in Handheld and ubiquitous computing. Springer, 1999, pp. 304–307.

[8] A. K. Dey, "Understanding and using context," Personal and ubiquitous computing, vol. 5, no. 1, 2001, pp. 4–7.

[9] L. Barnard, J. S. Yi, J. A. Jacko, and A. Sears, "Capturing the effects of context on human performance in mobile computing systems," Personal and Ubiquitous Computing, vol. 11, no. 2, 2007, pp. 81–96.

[10] H. Korhonen, J. Arrasvuori, and K. Väänänen-Vainio-Mattila, "Analysing user experience of personal mobile products through contextual factors," in Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia. ACM, 2010, p. 11.

[11] L. Gorlenko and R. Merrick, "No wires attached: Usability challenges in the connected mobile world," IBM Systems Journal, vol. 42, no. 4, 2003, pp. 639–651.

[12] A. Oulasvirta, S. Tamminen, V. Roto, and J. Kuorelahti, "Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile hci," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2005, pp. 919–928.

[13] C. Kim, M. Mirusmonov, and I. Lee, "An empirical examination of factors influencing the intention to use mobile payment," Computers in Human Behavior, vol. 26, no. 3, 2010, pp. 310–322.

[14] T.-P. Liang and Y.-H. Yeh, "Effect of use contexts on the continuous use of mobile services: the case of mobile games," Personal and Ubiquitous Computing, vol. 15, no. 2, 2011, pp. 187–196.

[15] R. Ali, F. Dalpiaz, and P. Giorgini, "A goal-based framework for contextual requirements modeling and analysis," Requirements Engineering, vol. 15, no. 4, 2010, pp. 439–458.

[16] J. Krogstie, "Requirement engineering for mobile information systems," in Proceedings of the seventh international workshop on requirements engineering: Foundations for software quality (REFSQ01), 2001.

[17] E. Eshet and H. Bouwman, "Addressing the context of use in mobile computing: a survey on the state of the practice," Interacting with Computers, 2014, p. iwu002.

[18] M. Kakihara and C. Sorensen, "Mobility: An extended perspective," in System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on. IEEE, 2002, pp. 1756–1766.

[19] H. B.-L. Duh, G. C. Tan, and V. H.-h. Chen, "Usability evaluation for mobile device: a comparison of laboratory and field tests," in Proceedings of the 8th conference on Human-computer interaction with mobile devices and services. ACM, 2006, pp. 181–186.

[20] S. Kristoffersen and F. Ljungberg, "Mobile use of it," in the Proceedings of IRIS22, Jyväskylä, Finland. Citeseer, 1999.

[21] J. Kjeldskov and J. Stage, "New techniques for usability evaluation of mobile systems," International journal of human-computer studies, vol. 60, no. 5, 2004, pp. 599–620.

[22] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," Network, IEEE, vol. 8, no. 5, 1994, pp. 22–32.

[23] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. IEEE, 1994, pp. 85–90.

[24] P. J. Brown, "The stick-e document: a framework for creating context-aware applications," Electronic publishing-chichester-, vol. 8, 1995, pp. 259–272.

[25] J. Pascoe, "Adding generic contextual capabilities to wearable computers," in Wearable Computers, 1998. Digest of Papers. Second International Symposium on. IEEE, 1998, pp. 92–99.

[26] T. Rodden, K. Cheverst, K. Davies, and A. Dix, "Exploiting context in hci design for mobile systems," in Workshop on human computer interaction with mobile devices, 1998, pp. 21–22.

[27] G. Chen, D. Kotz et al., "A survey of context-aware mobile computing research," Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, Tech. Rep., 2000.

[28] R. Harrison, D. Flood, and D. Duce, "Usability of mobile applications: literature review and rationale for a new usability model," Journal of Interaction Science, vol. 1, no. 1, 2013, pp. 1–16.

[29] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (ears)," in Requirements Engineering Conference, 2009. RE'09. 17th IEEE International. IEEE, 2009, pp. 317–322.

[30] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, "Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage," in Proceedings of the 13th international conference on Human computer interaction with mobile devices and services. ACM, 2011, pp. 47–56.

[31] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," IEEE Intelligent systems, no. 3, 1999, pp. 54–62.

[32] I. D. Constantiou, J. Damsgaard, and L. Knutsen, "The four incremental steps toward advanced mobile service adoption," Communications of the ACM, vol. 50, no. 6, 2007, pp. 51–55.

[33] Z. Zhang and X. Zheng, "User profiles and user requirements in mobile services," Perspectives in Business Information Research-BIR'2007, 2007, p. 170.

[34] Y. Liu and Z. Zhang, "Stakeholder-centered requirements elicitation: A view from user research," in 7th International Conference on Perspectives in Business Information Research, 2008, pp. 25–26.

[35] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on. IEEE, 2001, pp. 249–262.

[36] K. W. Thomas and B. A. Velthouse, "Cognitive elements of empowerment: An interpretive model of intrinsic task motivation," Academy of management review, vol. 15, no. 4, 1990, pp. 666–681.

[37] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," Advances in psychology, vol. 52, 1988, pp. 139–183.

[38] S. R. Barley, "On technology, time, and social order: Technically induced change in the temporal organization of radiological work," Making time: Ethnographies of high-technology organizations, 1988, pp. 123–169.

[39] S. Jumisko-Pyykkö and T. Vainio, "Framing the context of use for mobile hci," International Journal of Mobile Human Computer Interaction, vol. 2, no. 4, 2010, pp. 1–28.

[40] L. Chittaro, "Distinctive aspects of mobile interaction and their implications for the design of multimodal interfaces," Journal on Multimodal User Interfaces, vol. 3, no. 3, 2010, pp. 157–165.

[41] K. Schmidt and C. Simonee, "Coordination mechanisms: Towards a conceptual foundation of cscw systems design," Computer Supported Cooperative Work (CSCW), vol. 5, no. 2-3, 1996, pp. 155–200.

[42] F. Ljungberg and C. Sørensen, "Overload: From transaction to interaction," Planet Internet, 2000, pp. 113–136.

[43] Tencent Technology (Shenzhen) Company Limited, "Wechat." [Online]. Available: http://www.wechat.com/

[44] Google, Inc., "Gmail-email from google." [Online]. Available: https://itunes.apple.com/us/app/id422689480

[45] REDDIT, INC., "Alienblue-reddit official client." [Online]. Available: https://itunes.apple.com/us/app/id923187241

 ISBN: 978-1-61208-438-1

Design and Implementation of a Tool to Collect Data of a Smart City Through the TV

Glaydstone Alves Teixeira
CESAR – Recife Center for Advanced Studies and Systems
Centro de Estudos e Sistemas Avançados do Recife
Recife, Brazil
glaydstone.a.teixeira@gmail.com

Felipe Silva Ferraz
CESAR – Recife Center for Advanced Studies and Systems
Centro de Estudos e Sistemas Avançados do Recife
Recife, Brazil
fsf@cesar.org.br

***Abstract* — Smart City is the future of urban planning of the cities in the next generations. In Smart Cities, values are added to services, such as optimizing traffic and better use of energetic resources, generating a resource saving. These services are a result of a wide data collection from sensors or through crowdsensing. The objective of this work is to propose the development of a tool called PlugTV to collect information of citizens through the TV. The tool is composed by a Web component, implemented as a service, which allows data collection as a log from heterogeneous clients (TV, connected devices and sensors) and by a mobile component, implemented in an Android Mini-PC attached to the TV that allows data collection from the TV in a transparent way for the citizen, abstracting the dependency on a Digital TV middleware. In parallel, there is a gain of portability when it comes to IPTV providers because each Smart TV produces has its own platform of development and communication.**

Keywords- mini-pc; middleware; crowdsensing.

I. INTRODUCTION

Seventy percents of the world population, i.e., 7.2 billion people are expected to be living in cities and surrounding areas by the year 2050 [1][2][3]. Cities are having more control on their political, technological and economic development due to growing urbanization. Parallel to it, they face a series of challenges and threats to sustainability in all their central systems, which need to be approached in a holistic way [4].

Washburn and Sindhu [5] affirm that cities are becoming "smarter" once governments, companies and communities rely more and more on technology to overcome the challenges of fast urbanization. What makes a "smart city" is the combined use of systems, software infrastructure and a network of interconnected devices – the so called smart computing technology – in order to connect better seven components and infrastructure services of the city: administration, education, health, public safety, real estate, transportation and public services.

A Smart City is instrumented and tracked by a group of sensors and devices that collect data in such a way they can dynamically measure the urban activities of the city, helping the infrastructure of them [6]. This is the role of the smart software in smart cities. In this scenery, the paradigm of the Internet of Things (IoT) is based on the identification and use of a great number of physical and virtual objects disposed in a heterogeneous way and connected to the Internet.

IoT is now a subject of intense research. Technologies of the IoT are being implemented in a great number of applications. In the scope of Smart City initiatives, systems based on IoT are playing an important role, allowing the use of network infrastructures to introduce or improve a variety of services for the citizens. A network sensor is a key element in the Internet of Things, therefore, a key element in the Smart City applications. A variety of data is collected through sensors distributed all over the city or through crowdsensing. Data collected include typical information about the traffic, energy consumption of houses and apartments, devices connected to the Internet, etc. These data are stored as logs and analyzed by mining techniques, transforming the information into services that may be useful for the lives of citizens. However, the cost to implement and maintain these sensors is considerably high. Dohler and Ratti affirm that citizens will be the living sensors of smart cities and the central systems of the city should be connected to them [7].

This article proposes the development of a tool called ***PlugTV*** to collect data of the urban environment of a city and information about the routine of the citizen through the TV and provide the information collected as raw data into more accessible data, delivering information of much aggregated value.

This paper is structured as follows: Section II discusses state of the art. Our tool, its uses and functionalities are described in section III, while section IV describes technologies applied. Our studies and future works are presented in section V and section VI provides concluding remarks.

II. BACKGROUND

A. *Smart City*

Smart Cities are those that use advanced technologies to find solutions for their problems and for the new demands of

 ISBN: 978-1-61208-438-1

the population. What makes a city smart is the combined use of software, network infrastructure and client devices [6]. In the smart city, a variety of data is collected from sensors and people all over the city. These data include information about the traffic, energy consumption in houses and apartments, use of home appliances, etc.

Data are stored as logs and analyzed by advanced techniques of data processing, then used to aggregate value to services for a sustainable society.

According to Dirks and Keeling, the city is a like a system of systems [8]. No system works in an isolated way; instead, there is an interconnecting net. For example, transportation, industry and energy systems are closely related – transportation and industry are the main users of energy. Connecting these systems will offer more efficiency for the sustainability in the long run. The connection between the water and energy systems is another example of the connections there are between systems. A substantial amount of electricity generated goes to the pumping and treatment of water. In Malta, for example, a new smart utility system will inform citizens and companies about the use of energy and water, allowing them to make better decisions on the consumption of resources.

B. IOT

Internet of Things (IoT) and Cloud Computing are nowadays two of the most popular paradigms in Information and Communication Technologies (ICT) and should build the next computing era [9]. The IoT allows the communication between different objects, as well as the context of service innovation towards applications with greater aggregated value.

According to JIN et al [10], the network environment in IoT is strongly characterized by heterogeneity. Heterogeneous networks have a multi-service platform, providing different possibilities of services and applications. Cities are composed by a set of complex and heterogeneous systems of different kinds: infrastructure of civil engineering, ICT infrastructure, social media, financial networks, etc. All systems demand great management effort (tracking, reports and interventions) to guarantee the constant performance of relevant activities and services [11].

In 2008, the number of "things" connected to the Internet exceeded the number of people on Earth and by 2020 this number will exceed 50 billion things connected to the Internet, as seen in the following figure, presented by CISCO [12]:

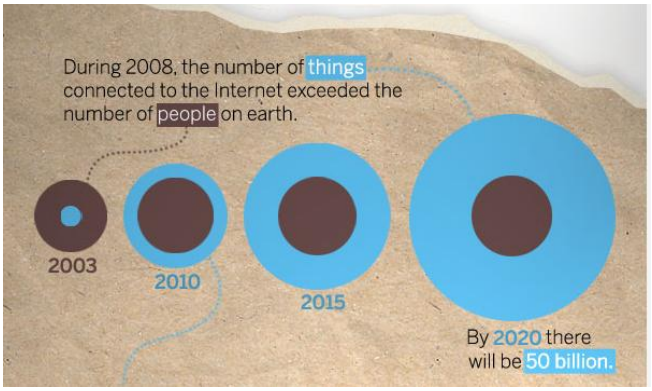


Figure 1. Growth of connected devices

As shown in Figure 2, the number of houses with TVs is three times higher in developing countries than in developed ones. When compared to the rest of the world, these numbers come to 1.3 higher. The discussed tool is extensible to treat heterogeneous data from different devices connected to the Internet, such as sensors, light, refrigerator, smart meters of water and energy, etc. The constant growth of the number of TVs in homes has motivated us to put the TV as the device used on this research for the proposed platform.

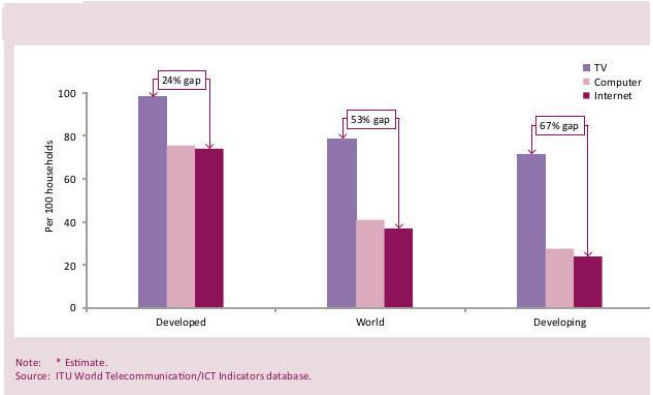


Figure 2. Growth of connected devices

C. Crowdsensing

Crowdsourcing is a new paradigm of managing information collection knowledge from a group of users, in order to execute complex tasks [13]. A well-known example of a crowdsourcing activity is Waze [14], one of the biggest traffic applications of the world based in a community.

III. PLUGTV

A. *Architecture of the PlugTV*

An application to collect data from the routine of the citizen through the TV is being developed. These data will be converted into useful service to the citizen. Figure 3 shows the general architecture of the PlugTV. Components of Figure 3 are the Smart City services, mobile component and WEB component, called PlugTVMobile and PlugTVRestful, respectively.

PlugTVMobile is implemented in an Android Mini-PC connected to the TV and it is responsible for collecting information from the routine of the citizen. When the TV is connected, the application creates an event via Broadcast notifying the change of status of the TV and this change is persisted through services disposed in the PlugTVRestful. Data will be converted into information such as: time when the citizen leaves home and arrives at work associated to services of the city as traffic information, climate catastrophes and flooding situations. As we can see, the architecture of the platform is composed by 4 components:

1) Architecture of the platform

a) Mini-PC (Gateway) – A Mini-PC installed in the house of the citizen. It is connected to heterogeneous devices connected to the Internet and it is through this gateway that some of the data processing happen to detect patterns on the routine of the citizen;

b) PlugTVRestful (WEB component) – It is a Web component implemented as a service in the Java EE 7 platform and it is organized as a RESSTful service, which has the Framework Jersey as its base. Loggers of the devices are persisted and sent in Json format through this component;

c) PlugTVMobile (Mobile component) – It is the mobile component developed in the Android platform and it is installed in the Mini-PC attached to the TV via HDMI to collect the logger. With the citizen properly registered and authenticated in the system, the device receives permission to send data to the Web platform and this component sends a logger automatically every time the TV is turned on, in an automatic and transparent way to the citizen, collecting data such as: time the TV was turned on, house and city where the device is being monitored, latitude and longitude, information about the city such as Gross Domestic Product (GDP), current manager, revenues, expenditures and territorial area;

d) *Smart City Services* – Available services of the cities, such as:

- Transportation.
- Healthcare.
- Education.
- Public safety and security.
- Building management.
- City administration.

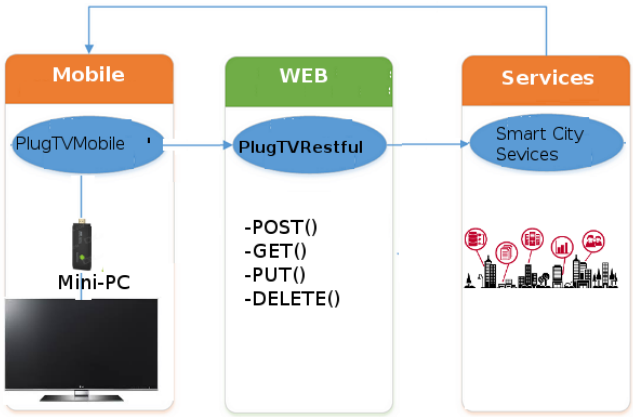


Figure 3. Architecture of the PlugTV.

B. *Characteristics of the PlugTV*

PlugTV is a proposal of developing software to collect and process data of the routine of the citizen through the TV. Because this a software, the development of the tool is based on some essential characteristics, such as: (a) reliability; (b) performance; (c) maintanability; (d) safety; (e) flexibility; (f) testing facility; (g) portability; (h) reusability; (i) interoperability.

Besides comprehending these characteristics, the PlugTV aims to eliminate two issues found in the development of applications for TV, which are: (a) dependency on the communication between the application and the Digital TV Middleware and how to order the application in the carrousel of data of the Digital TV providers; (b) existence of different platforms of application for Smart TV, since each manufacturer has their own SDK, which disables portability. With the use of PlugTV, these issues are abstracted, allowing an application to collect information through the TV and remaining in a RESTful server found in the clouds, without making these data go through the carrousel of data. PlugTV makes this application work in any Smart TV, regardless the producer and IPTV patterns.

Data will be collected every time the TV is turned on, in a transparent way for the user, so they can be analyzed and converted into information useful for the citizen, connecting them to services offered by the city, for example: traffic information, climate change and better use of resources, such as energy.

C. *Data Structure of the PlugTV*

PlugTV manages 2 types of data: house data, collected by devices (e.g., Smart TV) and settings data, which can be classified the following way:

1. **House settings:** Contains information about the house and has two entities: City Entity and House Entity;
2. **Device Settings:** Has two entities – Device and DeviceType. Device is the real device installed in the house and DeviceType classified the kind of device connected;

 ISBN: 978-1-61208-438-1

3. **Person Settings:** Has two entities: Citizen and Logger. Citizen represents all the citizen information and Logger contains information of all the devices of the citizen and routine data collected by the application.

Figure 4 illustrates the Diagram of Entity and Relationship of the PlugTV.

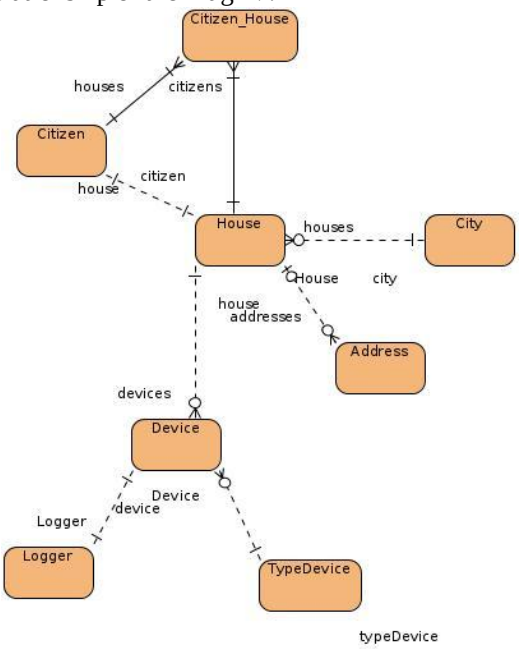


Figure 4. Design of the PlugTV database.

IV. TECHNOLOGIES APPLIED

PlugTV was developed in two modules: the one called ***PlugTVRestful*** was developed in the Java EE7 Platform and it is disposed as a ***RESTful*** server, with a Jersey Framework as base; the second module was implemented in the Android platform, running in a Mini PC attached to the TV. In order to manipulate the exchange of information with the server, the API JSON was used.

The main objective of the following sub-items is presenting artifacts generated during the implementation of the PlugTV.

A. *Functional Requirements*

The main functional requirements of the ***PlugTV*** are: (a) generating a routine logger of the citizen through the TV; (b) visualizing the collected data; (c) exporting the collected data in different formats, for example: xml, *json*, html, text/plain; (d) automatic saving of the TV logger.

The automatic saving of the TV log consists in handling data that will remain on the Restful server, called ***PlugTVRestful***. If the service that collects loggers is not available, data will be kept in the device to be synchronized after in the server.

B. *Non-functional Requirements*

Non-functional requirements refer to general features of the system, such as: safety, performance, distribution, maintainability and others. For the PlugTV, the following requirements were defined as non-functional: (a) Implementation using JAVA EE 7 to ensure portability; (b) Implementation using Android Version 4; (c) All API traffic should be over a secure connection using SSL; (d) Retrieved address and object information must be authenticated; Implementation of PlugTV Services

In order to provide services to be associated to services of smart cities, it was necessary to implement a services API of general use, including consultations to many data requests of the PlugTV. First, we are going to classify the types of service and their respective responses to requests. Services of the PlugTV were implemented as RESTful Web services and will be described after. Figure 5 presents the services and resources of the PlugTV.

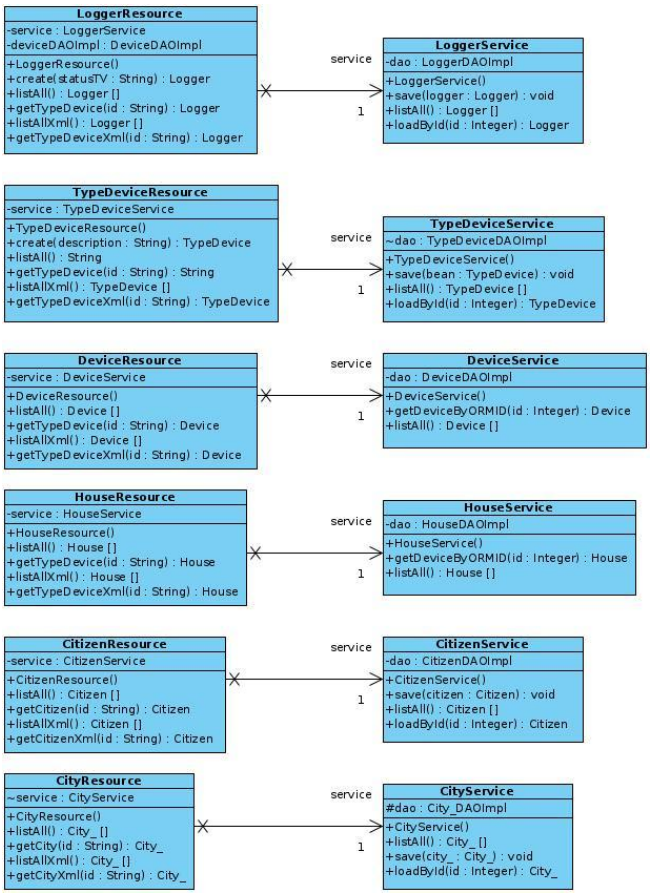


Figure 5. *PlugTV* services.

V. EXPERIMENTAL EVALUATION

This section discusses the method used to evaluate the proposed tool. In order to perform this analysis, two experiments were made, contextualized as tests. Two

servers were used to conduct the tests: the first representing the devices of the clients and the second representing the server, called **SERVER1** and **SERVER2** respectively.

SERVER1 was used to simulate users (TV) who send requests through the PlugTV tool. When it comes to the simulation of users, tests were recreated in the JMeter tool, version 2.3, a tool used to test the load in services offered by computer systems. This tool is part of the Jakarta project from the Apache Software Foundation. The computer called **SERVER2**, located in clouds, was responsible for hosting the web module **PlugTV** and process the requests sent by **SERVER1**.

Two tests called **Test1** and **Test2**, respectively, were performed to contextualize the experiments. The first simulated requests only where no event from the city was set by the tool, for example, a traffic jam at the time the citizen eventually leaves the house. In the second test, for every request made to the tool, an event of the city was associated. The metrics selected to track the performance of the tests were: *Times Over Time, Response Times Percentiles, Aggregate Graph.*

The setting of the requests of the users was made through Http Request the GET kind and configured as follows: Connect Timeout – 10.000(ms), Response Timeout = 10.000(ms). That is, if the request takes between zero and ten seconds to be opened or more than 10 seconds to be answered, this request is considered a defected one. The Path defined for the GET method described before follows the following format: /PlugTVRestful/logger/get/[request id]/.

A. *Times Over Time*

The objective of this metric is to measure the average time a request takes to be answered. That is, the time taken or the average delay between the start of a transaction and the results of it. These metrics are referenced and represented in JMeter by Times Over Time and Aggregate Graph.

In **Test1**, the user only sends his requests to module PlugTVRestful, which registers the logger of the TV. Since no event of the city was notified, an archive in the JSON format informs the logger was registered, unlike **Test2**. There, each request is associated to an event of the city, which besides sending the logger of the TV, an event associated to services of the city is signed, for example, traffic information, generating a JSON archive with information of the event.

In order to understand better the response time, the response time of each test is shown in Figure 6.

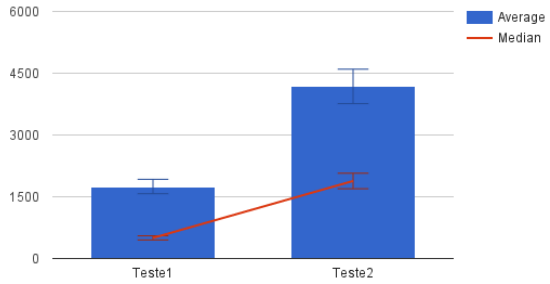


Figure 6. Response Time.

Percentiles are measures that divide the ordered sample (by ascending data order) in 100 parts, each one with a data percentage approximately equal. The k-nth Pk percentile is the x value (xk), which corresponds to the cumulative frequency of N k/100, where N is the sample size. Equation (1) demonstrates how the percentiles are calculated. The percentiles of the response time for each test are described in Figure 7. With an average response time of 1.749,4ms, Test1 is in a percentile of 79,5%. It means that 79,5% of the requests have a shorter or equal time than/to the average. Similarly, Test2 is in the percentile of 69,2%, with an average response time of 4.180,3 ms.

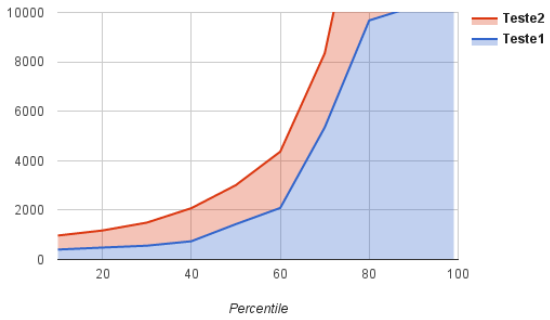


Figure 7. Percentile rank.

$$PR_x = \frac{\sum fb_x + \frac{f_x}{2}}{N}(100) \quad (1)$$

B. *Success and Failure Rate*

The objective of this metric is to analyze the success and failure behavior in the requests measures in the tests.

In order to perform the tests in JMeter, a metric called Transactions per Second was used, where the requests were set to suppose there was a failure in case of the connection

takes more than 10 seconds to be opened or more than 10 seconds to be answered.

For Test1, 122,7 requests were successfully processed and 6,76 requests per second failed. For Test2, 114, 8 were successful and 23,20 failures, as the following figure 8 shows.

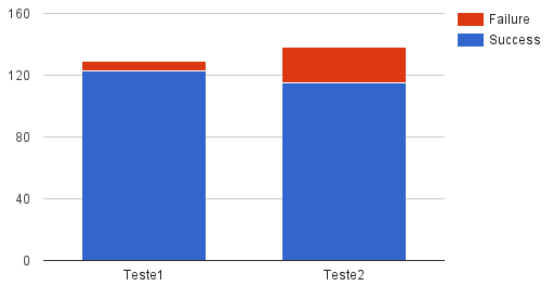


Figure 8. Success and Failure Rate.

VI. CONCLUSION

This paper proposed the development of a platform to integrate data of the urban space of a city with information on the routine of the citizen through a TV, in the context of the Internet of Things. The architecture is based on services and its objective is to provide information, abstracting the dependency of the communication of the TV with the Digital TV Middleware. For future works, the development and refining of the platform components, unit, integration and functional testing are expected. Another open question is increasing the number of devices connected to the tool, allowing a larger number of data. When it comes to data analysis, the development of a component that allows real-time analysis through techniques of Complex Events Processing (CEP) is desired.

REFERENCES

[1] J. Belissent, “Getting clever about smart cities: new opportunities require new business models,” 2010. [Online]. Available: https://www.forrester.com/Getting Clever About Smart Cities New Opportunities Require New Business Models/fulltext/-/E-res56701

[2] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami, “Network architecture and QoS issues in the internet of things for a smart city,” 2012 Int. Symp. Commun. Inf. Technol., pp. 956–961, Oct. 2012.

[3] Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N. M., & Nelson, L. E., Helping CIOs Understand “Smart City” Initiatives: Defining the Smart City, Its Drivers, and the Role of the CIO.Cambridge, MA: 2010 Forrester Research, Inc. Available at http://public.dhe.ibm.com/partnerworld/pub/smb/smarterplanet/forr_help_cios_und_smart_city_initiatives.pdf.

[4] Dirks, S., Gurdgiev, C., & Keeling, M. (2010). Smarter Cities for Smarter Growth: How Cities Can Optimize Their Systems for the Talent-Based Economy. Somers, NY: IBM Global Business Services. Somers, NY: IBM Global Business Services. Available at ftp://public.dhe.ibm.com/common/ssi/ecm/en/gbe03348usen/GBE03348USEN.PDF

[5] Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N. M., & Nelson, L. E. (2010). Helping CIOs Understand "Smart City" Initiatives: Defining the Smart City, Its Drivers, and the Role of the CIO. Cambridge, MA: Forrester Research, Inc. Available from http://public.dhe.ibm.com/partnerworld/pub/smb/smarterplanet/forr_help_cios_und_smart_city_initiatives.pdf.

[6] S. Nanni and G. Mazzini, “Smart City, a model and an architecture of a real project: SensorNet,” in Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on, 2013, pp. 1–4

[7] F. G. Dohler M., Ratti C., Paraszczak J., “Mart ities,” no. June, pp. 70–71, 2013.

[8] S. Dirks and M. Keeling, “A vision of smarter cities,” New York IBM Glob. Serv., p. 18, 2009.

[9] G. Suciu, A. Vulpe, S. Halunga, O. Fratu, G. Todoran, and V. Suciu, “Smart Cities Built on Resilient Cloud Computing and Secure Internet of Things,” in Control Systems and Computer Science (CSCS), 2013 19th International Conference on, 2013, pp. 513–518.

[10] D. Jin, Y. Zheng, H. Zhu, D. M. Nicol, and L. Winterrowd, “Virtual Time Integration of Emulation and Parallel Simulation,” in Principles of Advanced and Distributed Simulation (PADS), 2012 ACM/IEEE/SCS 26th Workshop on, 2012, pp. 201–210.

[11] E. Theodoridis, G. Mylonas, and I. Chatzigiannakis, “Developing an IoT Smart City framework,” in Information, Intelligence, Systems and Applications (IISA), 2013 Fourth International Conference on, 2013, pp. 1–6.

[12] CISCO, The Internet of Things, Infographic, available online at http://blogs.cisco.com/news/the-internet-of-things-infographic, 2011

[13] E. Aubry, T. Silverston, A. Lahmadi, and O. Festor, “CrowdOut: A mobile crowdsourcing service for road safety in digital cities,” in Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on, 2014, pp. 86–91.

[14] Waze (Community-Based Traffic and Navigation Application). Retrieved June 29, 2015 from https://www.waze.com.

[15] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

 ISBN: 978-1-61208-438-1

Comparison of Educational Project Management Tools

Rafael Queiroz Gonçalves, Christiane Gresse von Wangenheim
Department of Informatics and Statistics, Graduate Program on Computer Science
Federal University of Santa Catarina, UFSC
Florianópolis, Brazil
e-mail: rafael.queiroz@posgrad.ufsc.br, c.wangenheim@ufsc.br

***Abstract*—Project management tools are mandatory to properly manage software projects. The teaching of the usage of these tools is carried out in higher education computer courses and, usually generic tools are adopted, such as MS-Project. However, their lack of educational features has motivated the development of several educational project management tools. This study aims at the analysis of such existing tools, carrying out a systematic comparison. Therefore, we selected the most relevant educational project management tools based on the results of a Systematic Literature Review. These results were updated, including newly available tools and excluding proprietary and no longer available ones. The selected tools are presented, highlighting their educational features, supported functionalities and content coverage considering the whole project management process. A systematic comparison is conducted, discussing each evaluation criteria, resulting in a guideline for choosing the proper educational project management tool according to the educational goal. The presented results may be useful for instructors of Project Management courses as well as for researchers, to guide further research based on the identified gaps in this area.**

Keywords-Project Management; Project Management Tool; PMBOK; Teaching; Education; Open-source.

I. INTRODUCTION

Project Management (PM) is a critical area for many organizations in the software industry. A significant amount of projects still fail due to a lack of proper management, causing problems related to unaccomplished deadlines, budget overrun, or scope coverage [1]. In this context a project is considered a temporary endeavor to achieve a single result, and PM is the use of knowledge, abilities, tools, and techniques that enable a project to reach its goals [2].

Project problems occur mainly because of the absence of a PM process [3], resulting in a limited control over project restrictions and resources [1]. The adoption of a PM process may be facilitated by the usage of a PM tool [4]. A PM tool is a software that supports the PM process (either as a whole or partly), offering functionalities like: schedule development, resources allocation, cost planning, among others [7]. Despite the fact that many organizations still do not adopt any PM tool, the positive contributions that these tools may bring have increased the interest in their usage [5].

The responsibility for the usage of these tools lies with the project manager, who is accountable for the success of the project, having the authority to direct its resources in order to conduct the project by following a systematic PM process [2].

Given that the usage of PM tools is not yet common in organizations and that many projects still fail, a possible cause for this could be the lack of teaching the usage of these tools to project managers and team members [1][6][7].

The teaching of PM has to cover knowledge on PM, beyond general knowledge on administration, project environment, application area, and interpersonal abilities [2]. However, the teaching of PM should not just be focused on theoretical knowledge, as this is not enough for an effective PM application. And, as due to the complexity of contemporary software projects, PM is impracticable without the support of a PM tool, and the ability to use such tools is also among the project manager's competencies [4][8].

In Section 2, we present the background Section, followed by Section 3 that presents the analysis of related studies that have compared PM tools. In Section 4 we present the process we have adopted to carry out the educational PM tools comparison. In Section 5 we present each of the selected PM tools, and a structured comparison is presented in Section 6, leading to a discussion about each evaluated criteria in Section 7, resulting in a guideline for choosing the proper educational PM tool according to the educational goal. In Section 8 we present the conclusions of this study. These results may assist teachers in the teaching of this competence. They may also assist researchers in the improvement of support to the existing tools, or the development of new ones, covering the gaps that remain in this area.

II. BACKGROUND

Concepts that are relevant to this research are presented in this Section, namely: PM, PM tools, and teaching of PM tools. These concepts are used during the discussion of our findings, in terms of criteria for selection and evaluation of educational PM tools, or for analyzing their educational characteristics and general functionalities.

A. Project Management

PM conducts project activities and resources to meet its requirements, from its initiation to closure (Figure 1).

 ISBN: 978-1-61208-438-1

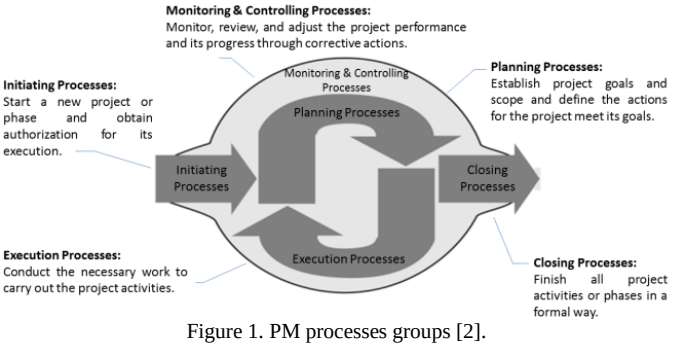


Figure 1. PM processes groups [2].

Orthogonally to these process groups, the PM processes are organized in 10 knowledge areas (TABLE I).

TABLE I. PM KNOWLEDGE AREAS [2].

Knowledge area	Processes to:
Integration	Identify, define, combine, unify, and coordinate PM processes and PM activities.
Scope	Ensure that the project addresses the entire work and meets all its requirements.
Time	Plan, monitor and control the activities that will be carried out during the project so it concludes within the deadline.
Cost	Plan, estimate, and control project costs, so it concludes within the approved budget.
Quality	Define the responsibilities, goals, and quality policies so the project meets the needs that have initiated it.
Human Resources	Organize and manage the project team.
Communication	Ensure the generation, collection, distribution, storage, recovery, and final destination of project information.
Risk	Identify, monitor and control the project risks.
Acquisition	Buy or contract products, services or any resources that are not available as project internal resources.
Stakeholder	Identify and manage the stakeholders and its expectations.

In the context of this study, the PM process refers to the one defined by PMBOK [3], which is the main reference in this area and widely accepted [9].

The application of a PM process is assisted by the usage of PM tools, which support the PM process, either as a whole, or a particular part of it. This support may semi-automatize some activities of PM process, such as writing status reports or providing online forms to record meeting minutes [6]. Furthermore, some PM process activities may be totally automated by PM tools, such as for instance, calculating the total project cost, the identification of the critical path, or the identification of over-allocated resources [5][10].

B. PM Tools

Conducting the PM process may be very complex and demand considerable resources of an organization. To assist in its execution, many PM tools have been developed. Examples include: MS-Project, Primavera, DotProject, Project.net, etc. [4][11]. However, due to the wide variety of PM tools, their functionalities and characteristics are very heterogenic [5, 12]. Supported functionalities, for instance, may cover the whole PM process, or just one or a few PM knowledge areas, or, more specifically, just some activities, for example, the tracking of work hours [11, 12]. The scope of the offered functionalities influences the usage of these tools for teaching, as they may restrict the addressed content.

Beyond its functionalities, other characteristics may also influence the choice of a PM tool to be adopted for teaching. According to its characteristics, such a tool may require some particularities in the computational environment besides the need for economic investments. Among these characteristics are: availability, platform, and usage propose.

The availability of PM tools may be proprietary (the use of a license or acquisition is mandatory and it is maintained exclusively by a single organization) or open-source (free usage and maintained by users community). Consequently, proprietary PM tools may be adopted only by organizations that are prepared to acquire its licence, while others may prefer to adopt as more low-budget alternative open-source tools.

In terms of platform, there are available stand-alone tools (mono-user and accessed via desktop) or web-based systems (multi-user and accessed via web browser). In practice, a web-based PM tool has to be used in order to properly manage a software project, as they allow collaborative work and sharing of information [4][5]. Thus, the teaching of these tools prepares the student better for a professional career [5]. However, the adoption of a PM web-based tool for educational proposes requires that this tool is installed on a web server that complies with the tool specification, and the students must have internet access.

Beyond the generic PM tools, such as MS-Project or DotProject that are focused on the professional daily routine, there also exist educational PM tools, which focus on student learning, such as ProMES and PpcProject [10]. These tools include didactic features, such as instructions about the usage of its functionalities, and simulations that create scenarios that propitiate the application of specific PM techniques.

C. Teaching of PM Tools

The usage of PM tools is part of the project manager responsibilities [2]. The need for teaching this competency is addressed by the ACM/IEEE reference curriculum for Computer Science [13]. It specifies that students have to develop knowledge in all PM knowledge areas, and have to learn the usage of a PM tool to develop a project schedule, allocate resources, monitor the project, etc. Often the teaching of PM tools usage includes the application of the following techniques [2][7][10]: the Critical Path Method (CPM) – that identifies the project activities that cannot be delayed without affecting the project deadline; the Program Evaluation and Review Technique (PERT) – that calculates the estimated effort to carry out an activity based on three other estimates (worst case, most common case, and best case); the Responsibility assignment matrix (RACI Matrix) – that describes the participation by various roles in completing project activities; Resources Leveling – technique in which start and finish dates are adjusted based on resource constraints,

 ISBN: 978-1-61208-438-1

with the goal of balancing demand for resources with the available supply; amongst others.

III. RELATED STUDIES

Several studies have presented comparisons of PM tools using different criteria for tools selection and evaluation [5][11][14][15][16].

Mishra et al. [14] compared 20 popular PM tools, presenting a brief description of each one and comparing them, based on criteria like platform, availability, and functionalities (e.g., resources management, schedule development, and earned value analysis). However, no PM tools selection criteria were presented.

Dippelreiter et al. [15] presents the comparison of 4 popular open-source PM tools that are adopted in industry, but again do not present selection criteria. The evaluation criteria were based on a set of functional requirements obtained after conducting interviews with project managers. Among these functional requirements are: project maintenance, contacts, activities, costs, documents download/upload, etc.

Margea et al. [16] compares 9 PM tools, including proprietary and open-source, and also stand-alone and web-based PM tools. Selection criteria were not presented. This study presents a description of each tool, including its main features and functionalities. Then, these tools are compared based on their platform and supported functionalities (e.g., resource management, risk analysis, schedule development, etc.).

Cicibas et al. [5] presents a comparison of 10 PM tools, including proprietary and open-source tools, and stand-alone and web-based. They included tools that were subject of previous scientific studies, as well as to be popular in the PM community (based on forums, blogs, and non-official web sites).Besides these characteristics, the PM tools were compared based on their functionalities, including: schedule development, resource management, time tracking, change management, document management, risk assessment, collaboration, amongst others. These evaluation criteria are explained, describing the expected functionalities that characterize its attendance.

Pereira et al. [11] presents a comparison of open-source PM tools. These tools were selected based on a systematic search in Sourceforge, the most relevant repository of open-source tools, and the comparison criteria were based on a unified best practice of PMBOK [2] and CMMI-DEV [12]. This study has compared 5 PM tools, which are claimed to be the most relevant based on the defined criteria. For each PM tool the supported PM best practices are identified.

Analyzing these comparisons we may conclude that currently there exist a wide variety of PM tools, and although they share some common features, their functionalities vary significantly. Thus, the PM process is partially supported by most of these tools and the choice of a PM tool may differ according to organization demands. However, before choosing a PM tool, it is important to know how to use its functionalities to support the PM process, hence, performing a conscious choice. Aiming to assist in the teaching of PM tools functionalities, some educational PM tools were developed, but no comparison with this specific focus has been encountered. In this context, the contribution of the work presented here lies in the analysis and comparison of relevant open-source educational PM tools.

IV. TOOL ASSESSMENT

The goal of this work is to present relevant educational PM tools and to assess their characteristics, educational features and functionalities. To systematically carry out the tool assessment, we adopted the following research process:

(1) Selection of educational PM tools, based on previous researches that present these tools.

(2) Definition of evaluation criteria with respect to the PM tools characteristics, educational features, and general functionalities.

(3) Execution of the PM tools evaluation.

(4) Analysis and interpretation of the collected data.

This process has been conducted by a PhD student of the Graduate Program in Computer Science (PPGCC) of the Federal University of Santa Catarina/Brazil, and revised by a senior researcher with expertise in Software Process Improvement and Project Management.

A. Tools selection

Aiming at the selection of relevant educational PM tools, we based our selection on a previous research carried out by the authors, which performed a Systematic Literature Review on the teaching of the usage of PM tools [17]. Among the results of this study is the identification of educational PM tools adopted for teaching. In the current study we performed a deeper exploration of each PM tool, identifying when each of them may be adopted and creating a guideline to instructors, so they can choose which educational PM tool may be adopted according to their educational goals. Moreover, for this study we have updated the results found on [17], including new educational PM tools.

In this context, the inclusion criteria for tool selection are:

- PM tool must include educational features;
- PM tool must be open-source; and
- PM tool must be available for download.

The exclusion criteria are:

- The software must be a PM tool (not games, simulators or e-learning platforms); and
- The tool must be focused on “traditional” PM (e.g., excluding any tool focused exclusively on agile PM).

This search, conducted in June 2015, returned a total of 10 educational PM tools. Applying the defined inclusion and exclusion criteria, only 5 educational PM tools have been considered relevant in the context of our study. We excluded tools such as EduSet [18], CBT Module [19], and POM-QM [20], which appear to be no longer available. Other tools such as PSG [21] and PTB [22] were excluded because they are proprietary tools. These tools were excluded, as we aim at presenting only PM tools that currently may be adopted by instructors or researchers to assist in their activities. The selected educational PM tools are presented in TABLE II.

 ISBN: 978-1-61208-438-1

TABLE II. SELECTED EDUCATIONAL PM TOOL.

PM Tool	Available for download at:
DrProject [23]	www.drproject.org
PpcProject [10]	http://code.google.com/p/ppcproject/
ProMES [24]	www.simor.mech.ntua.gr/Kirytopoulos/promes.asp
DotProject+ [25]	http://www.gqs.ufsc.br/evolution-of-dotproject
RESCON [26]	http://www.econ.kuleuven.be/rescon/

B. *Evaluation criteria*

Considering that only open-source educational PM tools are being evaluated, the evaluation criteria include the platform (stand-alone or web-based), educational features (aggregating all variations of educational features presented by the evaluated tools), PM techniques (that contains some educational support), and PM process coverage (in terms of knowledge areas and processes groups). These criteria are presented in TABLE III.

TABLE III. EVALUATION CRITERIA.

Description	Items to be evaluated
Platform	Stand-alone or web-based.
Educational PM features	• Scenarios to assist the application of specific PM techniques. • Feedback when students make some wrong usage of PM tool. • Hints to guide the student in the usage of PM tool. • Problems to be solved and definition of difficulty level. • Instructional materials to assist in the learning of PM tool usage. • Communication channels between students and teacher
PM techniques	CPM, PERT, Resources Leveling, RACI Matrix.
PM process coverage	
Knowledge areas	Integration, scope, time, cost, quality, communication, human resource, risk, acquisition, stakeholders.
Process Groups	Initiation, planning, execution, monitoring & controlling, closing.

For the evaluation of these criteria, only the functionalities that were presented by the authors of the PM tools were considered, excluding any undocumented functionality or extensions that may have been developed after their publication.

V. RESULTS EVALUATION

In this Section, we analyze each selected tool. The information presented for each tool include: its objective (for what it was designed), platform, a screenshot, main functionalities and educational features.

A. *DrProject*

DrProject (Figure 2) is a web-based PM tool, which was designed to assist students to understand the concept of a project and its lifecycle. It includes functionalities that assist the students to carry out an entire software project with team work, from its initiation to closure. Its main functionalities include features to assist team work, such as wiki, tickets, documents repository and mailing list. In addition, it also contains functionalities for definition of project activities and milestones.

Figure 2. DrProject.

The educational feature of this tool provides the instructor with a view of how students are performing at intermediate milestones. The forms were optimized to contain a minimum set of fields needed for didactic purposes, making it easier for students to understand the tool usage. The tool also provides administrative features that make it easier for the teacher to setup new projects and create new groups every term, thus reducing the time the instructor has to spend with administrative duties.

B. *PpcProject*

PpcProject (Figure 3) is a stand-alone tool that was developed to assist in the teaching of a PM tool with respect to CPM, PERT, and resources leveling techniques. This tool also has the goal to be at least comparable by students with other generic and proprietary tools, such as MS-Project.

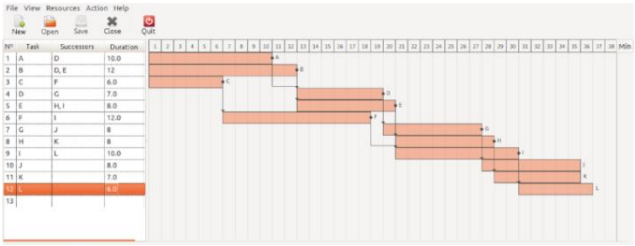


Figure 3. PpcProject.

The main functionality of this tool is focused on the schedule development, thus, supporting activity definition and sequencing, estimation of resources and durations, besides schedule development (Gantt chart).

The educational features are organized in three main modules: CPM, PERT, and Resource Allocation, which are the PM techniques this tool aims at teaching. The use of the CPM module is intended for students to deepen their understanding on the concepts of project activity decomposition, to analyze precedence relationships and to learn how to identify activities that cannot be delayed to achieve the expected completion date of the project, as well as to correctly interpret the Gantt chart. Using the PERT module students are expected to be able to calculate the project completion date in a probabilistic context and analyze the paths and critical activities during the project implementation. The Resources Allocation module includes features such as resources allocation, and identification of over allocated resources to apply resources leveling methods. By using this module, the students should be able to understand the influence of resource limitations on the project scheduling and propose alternative scheduling to improve resource usage.

 ISBN: 978-1-61208-438-1

C. ProMES

ProMES (Figure 4) is a stand-alone tool that was developed exclusively for academic purposes and aims at students to understand how CPM, PERT, and RACI are used, besides enhancing conditions for the acquisition of the required knowledge based on pedagogical approaches.

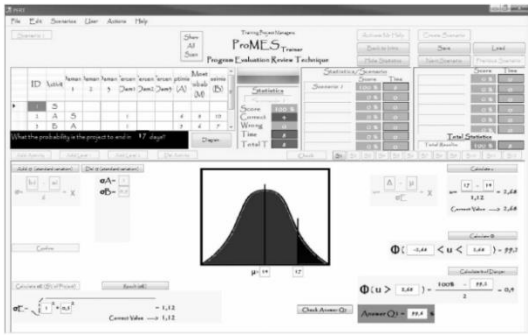


Figure 4. ProMES.

General functionalities include activity definition and sequencing, record estimations for effort, duration, and resources. The tool also supports the configuration of human resource roles and their allocation.

The educational features of ProMES, include CPM, PERT, and RACI matrix techniques. This tool offers the students feedback through interaction with the system. When the student begins to solve a scenario (exercise), the system checks and displays in message style all the errors. The student may revise his/her thoughts and try another solution. This procedure continues until no errors can be identified by the system. So, the student learns how to use the tool through feedback and tool interaction. Another very important educational aspect of ProMES is the help offered to the novice student. When the student first accesses the tool interface, a demonstration of how the tool works is displayed. In addition, the tool also gradually increases the difficulty of the proposed scenarios.

D. DotProject+

DotProject+ (Figure 5) has been developed to support the PM process to all knowledge areas for the initiation and planning processes groups. The educational goal of this tool is to assist the student to learn how to create a project charter and the project plan, supported by a PM tool. This tool is web-based and it is an enhancement of the generic PM tool – DotProject.

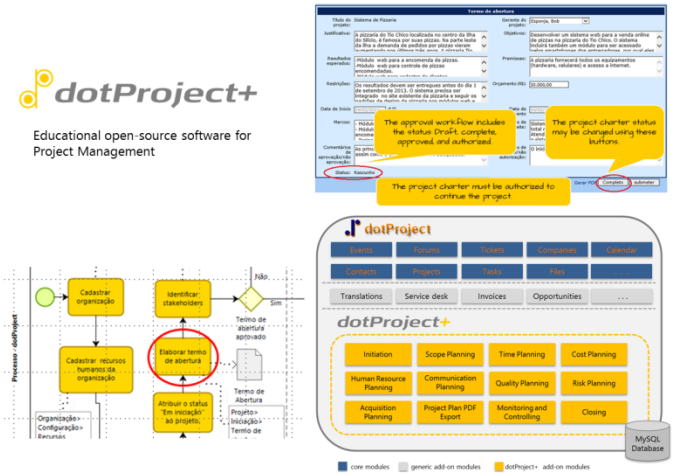


Figure 5. DotProject+.

Its functionalities include all standard functionalities of DotProject core modules, e.g., schedule development, calendar, contacts list, forum, tickets, etc. It also contains several add-on modules to include functionalities to cover all knowledge areas, for instance, registration of risk analysis, planned acquisitions, quality control plan, project stakeholders, etc.

Among the educational features, this tool includes instructional material, which explains the PM process and how it is supported by the tool's functionalities. Thus, it assists the student to conduct the PM process through learning the usage of a PM tool to support its execution.

E. RESCON: Educational Project Scheduling Software

The RESCON (Figure 6) is a stand-alone tool that focuses on the scheduling part of the PM process. It presents to students instances of the Resource-Constrained Project Scheduling Problem (RCPSP), that have to be solved with one of the many types of scheduling algorithms that are embedded in the educational PM tool.

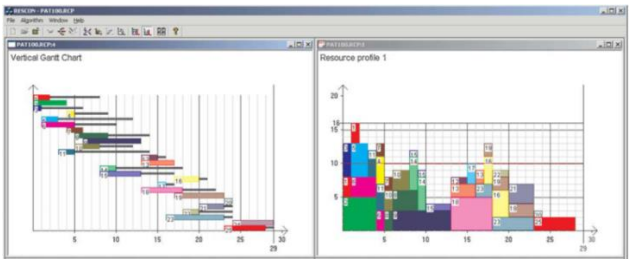


Figure 6. RESCON.

The general functionalities are related to the schedule development, and also with the human resource profiles configuration and allocation.

The educational features of this tool include the execution of CPM and the results of its execution are plotted with colored rectangles, which assist in the understanding of the related concepts, such as earliest and late possible start date, activity slacks and resources over allocation. The main educational feature of this tool lies in providing 48 kinds of

 ISBN: 978-1-61208-438-1

algorithms to schedule development that may be executed and their results compared.

VI. COMPARISON

As part of the evaluation, we compare the support provided by the PM tools in relation to each evaluation criteria as presented in TABLE III. For the criteria related to PM tools platform, PM techniques and educational feature we used a nominal rating scale, indicating whether the PM tool contains or not a certain feature. For the criteria related to the PM process coverage we used a 4-point ordinal rating scale, rating the support level for each knowledge area or processes group as presented in TABLE IV.

The results of the comparison of the educational PM tools are presented in TABLE V, where each PM tool is evaluated over the defined criteria (TABLE III) using the evaluation scales (TABLE IV).

TABLE IV. SCALES FOR RATING THE EVALUATION CRITERIA.

Nominal rating scale	
-	The tool does not contain the feature.
X	The tool contains the feature.
4-Points ordinal rating scale	
-	The tool does not support the knowledge area or process group.
*	The tool supports minimally the knowledge area or process group.
**	The tool has a wide support for the knowledge area or process group, but it is not complete.
***	The tool offers complete support for the knowledge area or process group.

TABLE V. COMPARISON OF EDUCATIONAL PROJECT MANAGEMENT TOOLS.

Evaluation Criteria / **Educational PM Tools**	**DrProject**	**PpcProject**	**ProMES**	**DotProject+**	**RESCON**
Platform					
Stand-alone	-	X	X	-	X
Web-based	X	-	-	X	-
PM techniques (with educational support)					
CPM	-	X	X	-	X
PERT	-	X	X	-	-
Resource Leveling	-	X	-	-	-
RACI Matrix	-	-	X	-	-
Educational features					
Scenarios to assist the application of specific PM techniques.	-	X	X	-	X
Feedback when students make wrong usage of PM tool.	-	X	X	-	-
Hints to guide the student in the usage of PM tool.	-	X	X	X	-
Problems to be solved and definition of difficulty level.	-	X	X	-	-
Instructional materials to assist in the learning of PM tool usage.	-	-	-	X	-
Communication channels between students and teacher	X	-	-	-	-
PM process coverage					
Knowledge Areas					
Integration	*	-	-	***	-
Scope	*	-	-	***	-
Time	*	***	***	***	***
Cost	-	-	-	***	-
Quality	-	-	-	**	-
Communication	**	-	-	**	-
Human resource	*	**	**	***	**
Risk	-	-	-	***	-
Acquisition	-	-	-	**	-
Stakeholder	**	-	-	***	-
Processes Groups					
Initiation	*	-	-	***	-
Planning	**	**	**	***	**
Execution	*	-	-	**	-
Monitoring and Controlling	*	-	-	**	-
Closing	*	-	-	***	-

 ISBN: 978-1-61208-438-1

VII. DISCUSSION

Analyzing the educational PM tools it is observed that only a few tools were developed with that propose, when compared with the wide variety of existing generic PM tools. Overall, we can observe that each tool is able to assist in the teaching for the purpose it was designed. However, considering the complete PM process, most tools have targeted only a specific part of this process and have included some educational feature to assist the students to understand this part and how it may be supported by PM tools functionalities.

Regarding the **educational features** presented by the analyzed tools, it was observed that these functionalities vary according to the educational goals. When the tool aims to teach the usage of a certain functionality, it presents instructions to demonstrate how to operate the software. Some tools present these instructions dynamically, depending on the student interaction with the tool. Some tools also present some usage guide to demonstrate how and when the PM tool functionalities may be utilized. When the goal is to teach the project life cycle, the PM tool typically includes functionalities for students to carry out a project from its initiation to closure, providing communication mechanisms between team members and the teacher.

With respect to the adopted **platform**, it was observed that the tools, which focused on the teaching of PM process are web-based. This is due to the fact that such a platform is more suitable to the process application in organizations, supporting multi-user accesses and information sharing among project stakeholders. These educational PM tools are closer to the real environment the students are going to face on real life projects. Considering the stand-alone PM tools, we can observe that they give support to just a few PM techniques. The adoption of this kind of platform may be justified when demanding a more complex user interface, using many charts and dynamic interactions, which may be easier to be developed on such platform.

The **PM techniques** that are usually taught through educational PM tools are CPM, PERT, RACI matrix, and resources leveling. The CPM technique is taught through different approaches. The ProMES tool requests the students to identify the critical activities analyzing the activity precedence diagram. The PpcProject tool requests the students to identify the critical activities by calculating its floats. The PERT method is taught in a similar way by most tools, requesting the student to enter her/his estimations for project execution and the three scenarios (best case, most common, and worst case), and then presenting the calculation for the PERT method. The RACI matrix is taught by ProMES only**,** allowing the students to assign responsibility, accountability, and consultancy, as well as to designate the representative who must be informed for each project activity. The Resources leveling technique is explicitly taught only by PpcProject, which imposes limits to the allocation of a certain resource, with the student having to find alternatives to develop the project schedule within the defined constraints. Both RACI matrix and resources leveling are also supported by DotProject+, but in this tool the student learns how to use it by instructions contained in the tool usage guide.

In this context, it becomes evident that the part of the **PM process** mostly addressed includes the knowledge areas time and HR, especially for the planning process group. An exception is the PM tool DotProject+, in which the PM process support stands out from the others, being is an enhancement of one of most popular open-source generic tools for PM. Hence, its functionalities are inherited, and gaps were filled by the addition of add-on modules. Some other tools also present specific exceptions, for instance, DrProject has a wide support for project communication, including support for document repository, wiki, and mailing lists, and also presents at least a minimum support to all processes groups.

Based on this analysis and comparison, we may identify when each of these PM tools may be more suitable for teaching. In situations when the educational goal is related to the teaching of specific PM techniques such as CPM, PpcProject, ProMES, or RESCON may be adopted. These tools provide scenarios that make it possible for the students to apply these techniques, and also evaluate the correctness of their answers, besides providing some feedback and explanations that make the teaching of these techniques easier. Specifically for teaching the PERT technique, both PpcProject and ProMES may be adopted. However, the resources allocation process is only partially supported by different tools. The PpcProject assists in the resource leveling technique, and ProMES assists in the creation of the RACI matrix. When intending to have support to all these techniques in a single tool, DotProject+ is an option. However, the instructor has to adopt some instructional method to teach these techniques, as even though DotProject+ supports all these functionalities, it does not contain many dedicated educational features to assist the students.

Nevertheless, when the educational goal is to teach about the PM process, DrProject and DotProject+ should be adopted. DotProject+ gives a wider support to the PM process than DrProject, providing the students with the opportunity to learn how to use several functionalities dedicated to all knowledge areas, such as risks, acquisition, quality and others, which are not covered completely by DrProject. Thus, it is important to understand that each educational PM tool has its own purpose. But the complexity of the usage of a PM tool is still not completely covered by any of the analyzed tools. So, it is important to know which tool may be suitable for each situation, according to the educational goals.

A. Threats to Validity

Several potential threats may have reduced the validity of the results of our work. The first threat we have identified is the existence of educational PM tools, which were not found by this research. This may occur when some PM tool was developed, but not published. However, we minimized this risk by basing our search on earlier and related work and performing an broad search on this topic.

Another threat to this research is the personal opinion of the authors by extracting the evaluated characteristics. To minimize this risk we considered only functionalities and

 ISBN: 978-1-61208-438-1

educational features that were documented. Thus, it is possible that some PM tool contains some additional feature or functionality that is not documented, which thus was not included in our analysis.

Our specific focus on open-source educational PM tools may also represent a threat. As a consequence we excluded 2 proprietary tools, PSG and PTB, which claim to have many interesting educational features, such as simulations, that propitiate the student to have ideal scenarios to learn about project monitoring and controlling. However, these tools are not fully available to be adopted by instructors or researches without additional expenses.

VIII. CONCLUSIONS

This research aims at the comparison of educational PM tools that are openly available, identifying their characteristics, educational features, and supported functionalities. To reach this goal, educational PM tools were selected based on results derived from a Systematic Literature Review. Each selected PM tool was analyzed. The analysis describes each evaluated tool and presents a discussion of when each of these tools is more suitable to be adopted for teaching.

Hence, despite the efforts, it is evidenced that there is no tool that is complete enough to attend all educational demands, and it still is necessary to adopt a set of tools, according to the part of the PM process it aims at teaching. Future work may suggest enhancements on some educational PM tool to include support to all knowledge areas and process groups, expanding the educational features and then covering the gaps still existing in the teaching of the usage of PM tools.

ACKNOWLEDGMENT

This work was supported by the CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico – www.cnpq.br*), an entity of the Brazilian government focused on scientific and technological development.

REFERENCES

[1] The Standish Group, Chaos Manifesto 2013, Boston, 2013.

[2] PMI – Project Management Institute, A Guide to the Project Management Body of Knowledge, 5. ed., Newtown Square, 2013.

[3] M. Keil, A. Rai, and J. Mann, “Why software projects escalate: The importance of project management constructs,” IEEE Transactions on Engineering Management, vol. 50, n.3, 2003, pp. 251–261.

[4] R. Fabac, D. Radoševic, and I. Pihir, “Frequency of use and importance of software tools in project management practice in Croatia,” In: Proc. of 32nd Int. Conf. on Information Technology Interfaces, Cavtat, 2010, pp. 465 -470.

[5] H. Cicibas, O. Unal, and K. Demir, “A comparison of project management software tools (PMST),” In: Proc. of the 9th Software Engineering Research and Practice, Las Vegas, 2010.

[6] T. Lethbridge, J. Diaz-Herrera, R. Leblanc, and J. Thompson, “Improving software practice through education: Challenges and future trends,” In: Proc. of Future of Software Engineering, Minneapolis, 2007, pp. 12-28.

[7] Ž. Car, H. Belani, and K. Pripužić, “Teaching Project Management in Academic ICT Environments,” In: Proc. of the Int. Conf. on computer as a tool, Warsaw, 2007, pp. 2403 - 2409.

[8] L. Spencer, and S. Spencer, Competence at Work: Models for Superior Performance, 1st ed. John Wiley & Sons, 1993.

[9] O. Ojeda, and P. Reusch, "Sustainable procurement - Extending project procurement concepts and processes based on PMBOK," In: Proc. of 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Berlin/Germany, 2013, pp. 530 – 536.

[10] L. Salas-Morera, A. Arauzo-Azofra, and L. García-Hernández, “PpcProject: An educational tool for software project management,” Computers & Education, vol. 69, n. 1, 2013, pp. 181-188.

[11] A. Pereira, R. Gonçalves, and C. Wangenheim, “Comparison of open source tools for project management,” International Journal of Software Engineering and Knowledge Engineering, vol. 23, n. 2, 2013, pp. 189-209.

[12] C. Wangenheim, J. Hauck, and A. Wangenheim, “Enhancing open source software in alignment with CMMI-DEV,” IEEE Software, vol. 26, n. 2, 2009, pp. 59-67.

[13] ACM, and IEEE Computer Society, Computer Science Curricula 2013, 2013.

[14] A. Mishra, and D. Mishra, “Software Project Management Tools: A Brief Comparative View,” ACM SIGSOFT Software Engineering Notes, 38 (3), 2013, pp. 1-4.

[15] B. Dippelreiter. “A ‘state of the art’ Evaluation of PM – Systems exploring their missing Functionalities,” In: Proc. of the 5th Int. Conf. on Project Management", Tokyo, 2010, pp. 90-101.

[16] R. Margea and C. Margea. “Open Source Approach to Project Management Tools.Informatica Economică”, vol. 15, n. 1, 2011, pp. 196-206.

[17] R. Gonçalves, and C. Wangenheim. “How to Teach the Usage of Project Management Tools in Computer Courses: A Systematic Literature Review,” In: Proc. of the Int. Conf. on Software Engineering and Knowledge Engineering, Pittsburgh, 2015, pp. 36 - 41.

[18] J. C. Spicer. “A spiral approach to software engineering project management education,” ACM Sigsoft Software Engineering Notes 8(3), 1983, pp. 30–38.

[19] D. Pfahl, M. Klemm, and G. Ruhe, “A CBT module with integrated simulation component for software project management education and training,” Journal of Systems and Software, vol. 59, n. 3, 2011, pp. 283–298.

[20] H. Ku, R. Fulcher, and W. Xiang, “Using computer software packages to enhance the teaching of engineering management science: Part 1 - Critical path networks,” Computer Applications in Engineering Education, vol. 19, n. 1, 2011, pp. 26-39.

[21] M. Vanhoucke. “The Project Scheduling Game (PSG): Simulating Time/Cost Trade-Offs,” In Projects. Project Management Institute, vol. 36, n. 1, 2005, pp.51-59.

[22] A. Shtub. “Project management simulation with ptb project team builder,” Proceedings of the 2010 Winter Simulation Conference, Baltimore, 2010, pp. 242-253.

[23] K. Reid, and G.Wilson. “DrProject: A Software Project Management Portal to Meet Educational Needs,” In: Proc. of the Special Interest Group on Computer Science Education, Covington, 2007, pp. 317-321.

[24] G. Gregoriou, K.Kirytopoulos, and C. Kiriklidis, “Project Management Educational Software (ProMES),” Computer Applications in Engineering Education, vol. 21, n. 1, 2010, pp. 46–59.

[25] R. Gonçalves,. E. Kühlkamp, and C. Wangenheim. “Enhancing dotProject to Support Risk Management Aligned with PMBOK in the Context of SMEs,” International Journal of Information Technology Project Management, vol. 6, n. 2, 2015, pp. 40-60.

[26] F. Deblaere, E. Demeulemeester, and W. Herroelen. “RESCON: Educational project scheduling software,” Computer Applications in Engineering Education, vol. 19, n. 1, 2009, pp.327-336.

 ISBN: 978-1-61208-438-1

Model-Driven Engineering of Software Solutions for QoS Management in Real-Time DBMS

Salwa M'barek[a], Leila Baccouche[b], and Henda Ben Ghezala[c]
RIADI Laboratory
ENSI, Manouba University
Tunis, Tunisia
e-mail: [a]salwa.mbarek@gmail.com, [b]leila.baccouche@insat.rnu.tn, [c]hhbg.hhbg@gmail.com

***Abstract*—Real-Time applications handling big volumes of Real-Time data with time constraints, such as web-based multimedia applications or Vehicular Cyber-physical Systems, often lead to unpredictable overload problems in Real-Time DataBase Management Systems (RTDBMS) managing them. This is due to frequent Real-Time user transactions, requesting access to Real-Time data, which are characterized by unknown arrival times, unknown workloads and time constraints. In the literature, many software solutions with Quality of Service (QoS) management are proposed to resolve these problems. Although effective, they remain application-dependent regarding Real-Time data and transactions models and QoS requirements. Moreover, no formal or semi-formal models of these solutions or tools are proposed to design them. Therefore, it is not possible to reuse such solutions for Real-Time applications with specific QoS requirements and needing other data and transactions models. To address this issue, we propose a Model Driven Engineering based framework for modeling QoS management solutions in RTDBMS and reusing formal models of well-known solutions. The framework provides a tool and strategic methodology to help users achieve their goals through several strategies that fit their requirements.**

Keywords- Model Driven Engineering; QoS management; Real-Time DBMS; model transformations; reuse.

I. INTRODUCTION

Real-Time DataBase Management Systems (RTDBMS) are suitable to Real-Time Applications (RTA) handling a large volume of Real-Time data, which have validity periods and must be updated periodically to reflect the state of the application environment. They are able to manage Real-Time transactions, which are time-constrained (e.g., having deadlines) and frequently request access to Real-Time data [18]. Examples include the Vehicular Cyber-Physical Systems (VCPS) that must collect and handle a large volume of Real-Time data about vehicles and road traffic for ensuring road safety and providing various data services within accurate time deadlines [14]. There are also included web-based multimedia applications, such as video on demand, which manage large amounts of data and must respect the time constraints when transmitting video packets [12].

A RTDBMS executes periodic *update transactions*, which refresh Real-Time data to preserve the logical and temporal consistency of the Real-Time database. In addition, it manages transactions from users reading the Real-Time data, called *user transactions* and must meet their deadlines [8].

The workload of *update transactions* is known in advance, while the *user transactions* have unknown workloads and unpredictable arrival times [18]. Hence, the RTDBMS can face unpredictable overload periods and be no longer able to satisfy both types of transactions, which lead many Real-Time transactions to miss their deadlines. Thus, the RTDBMS stability may not be guaranteed [8].

In the literature, many QoS-aware solutions that we call QoS Management Solutions (QMS) aim to address this problem, such as the solutions proposed in [2][3][11][12]. The QoS guarantee is based on QoS requirements specified by the database administrator (DBA). Most QMS combine the Feedback Control Scheduling Architecture (FCSA) [16] with imprecise computation techniques [10], which allow graceful QoS degradation during transient overloads.

Although effective, these solutions are dependent on the Real-Time data and transactions models and requirements of the RTA. Moreover, we did not find in the literature formal or semi-formal models of these solutions or tools to model them. Therefore, it is not possible to reuse existing solutions for a RTA with specific transactions and data constraints or specific QoS requirements.

In this paper, we propose a model driven framework based on the Model Driven Engineering (MDE) approach [5] for modeling new QMS by reusing models of well-known solutions. The framework focuses on a strategic methodology to help users modeling QMS by and for reuse.

The rest of the paper is organized as follows. In Section 2, we present a state of the art on modeling works related to RT DBMS. Section 3 gives a brief presentation of the QoS management architecture in RTDBMS. The QMS components are the subject of Section 4. The framework architecture is detailed in Section 5. In Section 6, we present the proposed methodology. Section 7 presents the framework implementing. In Section 8, we give the results of experiments on a concrete QMS. We end this paper with a conclusion and give some future works.

 ISBN: 978-1-61208-438-1

II. RELATED WORK

QMSs mentioned above, that we have deeply studied, are not modeled. They are described in a natural language and presented through an architecture, QoS parameters, system variables and algorithms acting to meet the QoS specification. In the RTDMS domain, there are some object-oriented models focusing on Real-Time object-oriented databases modeling, and especially on system aspects, such as RTSORAC [6], RODAIN [21] and BeeHive [20].

These works propose object-oriented models that only consider structural and behavioral features of Real-Time data and transactions. Some of them partially support the modeling of scheduling, concurrency control, Real-Time data distribution and quality of service policies. They provide only specific data and transactions models and cannot support new data and transactions constraints.

Other works offer UML profiles for modeling RTDBMS, such as RTO-RTDB [15], RTO [9][13]. They provide UML extensions for Real-Time databases designers, through stereotypes, which express both the Real-Time data and transactions constraints.

The related work that we have presented provides a rich foundation upon which we can build. However, the QoS modeling is partially supported in these models, which provide specific QoS parameters and cannot specify new parameters. They are all focused on data management and do not consider transactions management. Moreover, there is no one work that focuses on QMS modeling and reuse.

III. QOS MANAGEMENT ARCHIRECTURE IN REAL-TIME DATABASE MANAGEMENT SYSTEMS

The QoS is specified by the DBA through *QoS parameters*, which are metrics defining the desired performance of the RTDBMS. The DBA specifies Quality of Data (QoD) and Quality of Transactions (QoT) parameters with reference values they must not exceed. In transient overloads, an overshoot of these thresholds may be tolerated by giving the worst-case system performance. Here, we give an example of QoS parameters proposed in [11].

- *Deadline Miss Ratio (MR)* is the percentage of user transactions that missed their deadlines regarding to the accepted ones. MR is considered as a QoT parameter.
- *Perceived freshness (PF)* is the ratio of fresh data to the entire temporal data in a database. Fresh data are data updated within their validity interval. It is considered as a QoD parameter.
- *Maximum Overshoot (Mp)* defines the worst-case system performance in the transient system state, e.g., the highest MR in the transient state. QoS parameters can overshoot their reference values at maximum of Mp.
- *Settling time (ts)* is the time for the transient overshoot to decay and reach the steady state performance.

In [11], authors propose the following QoS specification: $MR \leq 5\%$, $PF \geq 98\%$, $Mp \leq 30\%$ and $ts \leq 45$sec.

The well-known QMS for RTDBMS use the FCSA because it provides the QoS specification guarantee without a priori knowledge of transactions workloads using feedback control. FCSA has shown to be very effective for a large class of systems that exhibit unpredictable workload. For instance, it was applied to improve the QoS in distributed multimedia systems [22] and recently in geographic information Systems [23].

The basic FCSA applied by most QMSs in RTDMS (Figure 1) was proposed in [2]. This architecture is based on the principle of observation and self-adaptation. The observation is the periodic measurement of QoS parameters by a Monitor. The auto-adaptation is the dynamic adjustment

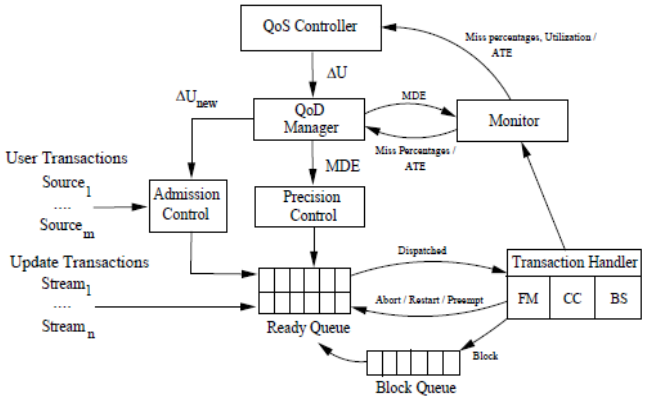


Figure 1. Basic Feedback Control Scheduling Architecture in RTDMS.

of the measured QoS to converge to the required QoS achieved by the QoS Manager. The adjustment is computed by a QoS controller containing feedback loops. The QoS Manager executes a QoS management algorithm, which implements adjustment scenarios based on system states [2].

The well-known QMSs in the literature [2][3][11][12] are efficient to guarantee the QoS specification even in transient overloads thanks to the FCSA and Imprecise Computation techniques, which allow the use of approximate results and imprecise data. However, these solutions may be inadequate for RTA requiring specific QoS requirements and using specific data and transactions models. It is more common to take a subset of their elements in order to reuse them for a new QMS design, which is not provided by related works.

IV. QOS MANAGEMENT SOLUTIONS COMPONENTS

Through the evaluation of the most referenced QMS [1], we conclude that their differences and similarities are focused on three components: *Models, Parameters* and *Policies*, which are interdependent. In addition, RTA requirements can be regrouped according to these components, which are worth of reuse to fit new requirements. For instance, take a RTA that requires a deadline miss ratio MR≤10% and its user transactions can follows the Milestone model that tolerates transaction

 ISBN: 978-1-61208-438-1

imprecision, but does not tolerate data imprecision for critical data. In this case, the suitable QMS must combine transactions and queues models from the FCS-IC1 QMS [2] with the data model, from the QMF1 QMS [11] and combine QoD parameters from QMF1 with QoT and loops parameters from FCS-IC1. Policies of QMF are suitable for this RTA with some reconfiguration. If the same RTA tolerates data imprecision, but requires specific QoS management scenarios, then the FCS-IC1 can be reused in this case, but its QoS management policy must be rewritten.

The component *Models* gathers the different models used by a solution, namely: the Real-Time transactions model, the Real-Time data model and the queues model, which vary from a solution to another. We employ the notion of data model to define data attributes, data types and data imprecision implementation. For instance, authors in [2] allow data to deviate, to a certain degree, from their corresponding values in the external environment. The transactions model denotes transactions attributes, transactions types and the transactions imprecision implementation, e.g., multiple versions, use of sieve functions and the Milestone approach [10]. The queues model describes the queues configuration through the number of queues, priority levels and types of transactions in each queue.

The component *Parameters* includes the QoD and QoT parameters, system parameters such as *ts* and *Mp* and feedback loops parameters that differ from one QMS to another. The *Parameters* component configuration depends on that of *Models* component.

The component *Policies* is based on the two previous components. It comprises algorithms which impact on the QoS, namely: the scheduling algorithm, the concurrency control algorithm, the updating algorithm and the QoS management algorithm. This later tries to balance the RTDBMS workload between user and update transactions through a compromise between QoT and QoD. For instance, if the system downgrades the QoD, many update transactions will be rejected and vice versa.

V. FRAMEWORK ARCHITECTURE

Our proposal is guided by the need to reuse some elements of QMSs in order to better meet new QoS requirement and specific constraints on Real-Time data and transactions. For this purpose, we provide to users a theoretical and practical framework allowing them design and extend their own QMS models. This framework is based on the reuse of the well-known QMS models, which are modeled by an expert using a specific editor and broken down into reusable fragments, that represents QMS components (Section IV).

A. Meta-modeling architecture

The framework provides "productive" QMS models that can be processed and transformed automatically in order to be reused. This is allowed using the OMG meta-modeling architecture [24] and models transformations [5], which are the core of the MDE approach that we adopted.

The proposed four-layered modeling architecture is based on the OMG meta-modeling architecture for the MDE (Figure 2). In the lowest layer M0 of systems, we find QMS. The layer M1 above M0 contains the models that represent QMSs, which we call QoS Management Approaches (QMA). A model in M1 must be conformant to its meta-model in the layer above, named M2, which defines all the concepts of a model and is considered as an abstract syntax of a specific modeling language to formalize model description. In this work, we designed a QMS meta-model named MM-SGQdS to formally express their corresponding QMA in a modular way that represents all QMS components. Thus, QMAs will be conformant to MM-SGQdS and can be easily manipulated and reused. To implement MM-SGQdS, we chose the EMF core (Ecore) meta-meta-model at the layer M3 (above M2) [4]. It is a subset of the OMG standard meta-modeling language MOF [17] that formalizes the description of meta-models in M2.

Let us now present the MM-SGQdS meta-model and its concepts.

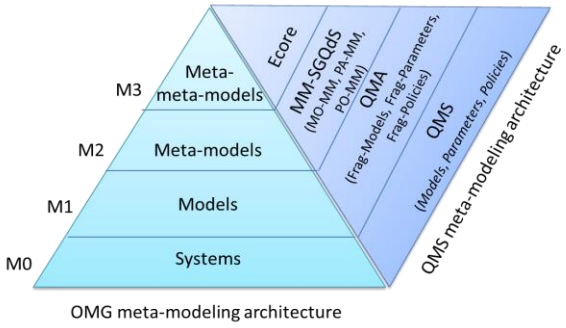


Figure 2. QoS Management Solutions Meta-modeling architecture.

B. QoS Management Solutions Meta-model

For each QMS component, namely: *Models*, *Parameters* and *Policies*, we proposed a specific meta-model (in layer M2) to formally express its elements and relationships between them. They are respectively called *MO-MM, PA-MM* and *PO-MM* (Figure 2). Thus, MM-SGQdS is comprised of these three meta-models as illustrated by Figure 3 showing its abstract view simplified.

MO-MM comprises three main meta-classes, namely: *TransactionsModel, DataModel* and *QueuesModel* describing elements of the *Models* component. PA-MM includes four main meta-classes, namely: *QoDParameters, QoTarameters, SystemParameters* and *LoopsParameters*

 ISBN: 978-1-61208-438-1

modeling elements of the *Parameters* component. PO-MM has four main meta-classes, namely:*QoSPolicy, SchedulingPolicy, UpdatingPolicy* and *ConcurrencyPolicy* correponding to elements of the *Policies* component. Each meta-model has a tree stucture with a root meta-class linking its main meta-classes. The root meta-classes of *MO-MM, PA-MM* and *PO-MM* are linked to a meta-class called *QoSApproach*, the root of the tree structure of MM-SGQdS.

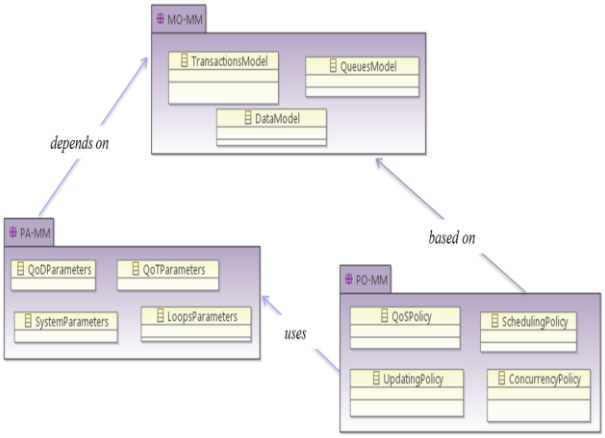


Figure 3. Abstract view of the meta-model MM-SGQdS.

A QMA conformant to MM-SGQdS is comprised of three reusable fragments called *Approach Fragments* (AF), namely: *Frag-Models, Frag-Parameters* and *Frag-Policies* (Figure 2), which represent respectively *Models*, *Parameters* and *Policies* component and conformant to *MO-MM, PA-MM and PO-MM* meta-model, respectively.

C. Logical architecture

In the software reuse domain, two kinds of actors are involved, namely: developers for reuse and developers by reuse. In this work, which focuses on QMA reuse, the framework involves two kinds of actors, namely the *RTDBMS experts* and the *QMA designers*. The *expert* generates and manages AF for reuse, while the *designer* reuses these fragments in order to build a new QMA. The two actors are guided by a methodology, which will be described later in this paper. Existing QMSs are modeled by the expert through a specific QMA editor to generate corresponding QMAs and then broken down into AF. The generated fragments are stored in a database to be reused by the designer or the expert in new situations (Figure 4).

For the decomposition of QMA, the expert uses a transformation called *Decomposition*, which generates the three AF. To build of a new QMA, the designer uses a transformation called *Composition*, which assemble AF from different QMA. The transformation called *Extension* is used to extend existing QMA by adding some elements from other approaches.

The new QMA are in turn broken down by the expert into AF using the *Decomposition* transformation.

Generated AFs are inserted into the database to be reused.

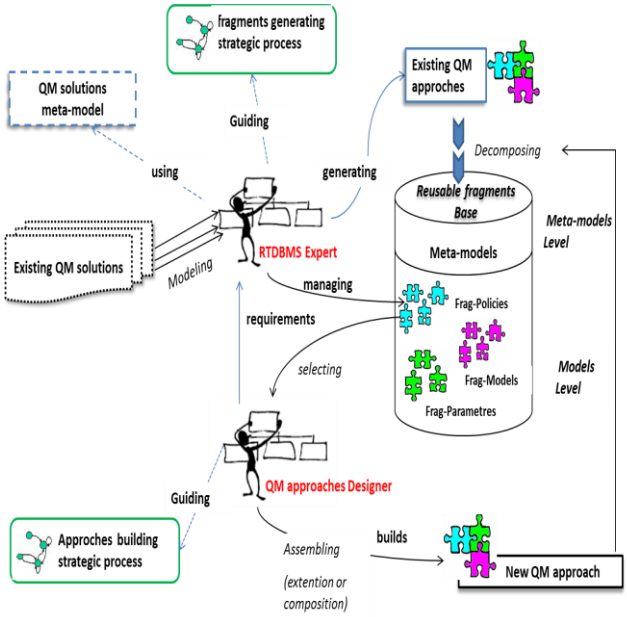


Figure 4. The framework logical architecture.

The two Processes in Figure 4 are part of a strategic methodology for QMA design, which is the subject of the following Section.

VI. Strategic Methodology for QoS Management Approaches Design

To guide framework actors to achieve their goals, we propose a methodology that defines two strategic processes, namely: GEN-PM, a process for generating AF (expert side) and CONS-PM, a process for building new QMA by reuse (designer side). These processes are modeled through the MAP formalism [19] for its flexibility and simplicity. A MAP-based process model, called *map*, is an oriented graph that represents explicitly goals to be achieved on nodes with the different strategies for achieving them on oriented labelled arcs. Subsequently, we detail the two processes models.

A. Approach fragments generating process

The map for modeling the AF generating process, named GEN-PM, contains different strategies related to three main goals, namely, *generating AF, managing AF* and *editing QMA* (Figure 5).

- *Generating AF*: three strategies are defined to achieve this goal. The strategy "*by decomposition*" aims to break down a QMA through a *Decomposition* transformation for QMA in order to extract the three AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*). Thereafter, generated AFs are stored in the database with their elements, which are generated through a *Decomposition* transformation for AF.

The strategy "*by extension*" aims to extend an existing fragment by adding it some elements with conformance to its meta-model. The expert can combine elements from different AF to build new AF, which constitutes the aim of the strategy "*by composition*". He can design a new AF "*from scratch*" for specific needs, using a specific AF editor provided by the framework.

- *Managing AF*: to achieve this intention, we set three strategies, namely: "*editing*", updates the values of an AF properties, "*deleting*", deletes un-usefull AF from the database, and "*storing*", adds a new AF in the database.
- *Editing QMA*: existing QMS are modeled through a specific editor based on their meta-model MM-SGQdS following the strategy "*by editing*".

The "*Start*" and "*End*" goals define the start and the end of the process, respectively. At the end of the process, the strategy "*validation*" is used to check the composition. Each new fragment must be stored in the database to be persistent and reused thereafter, which is the role of the strategy "*storage*".

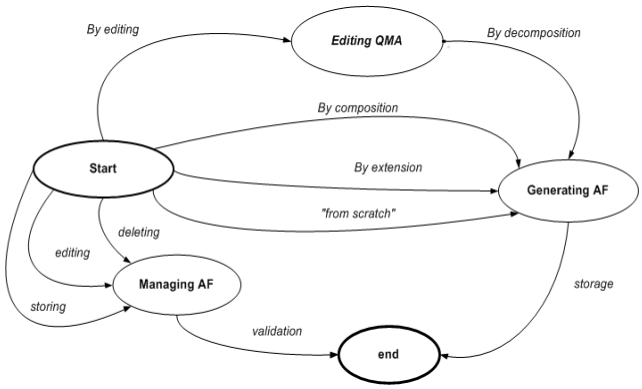


Figure 5. Map of approach fragments generating process.

B. QoS Management Approach building process

The process focusing on QMA building, called CONS-PM, is based on model reuse through model transformations. It represents designer goals and strategies to achieve them. The CONS-PM map comprises three major goals, namely *selecting AF*, *composing QMA* and *extending QMA*.

- *Selecting AF*: concerns the ways to search for AF from the database before composing them (by name, by type or by approach name).
- *Composing QMA*: builds a new QMA by linking AFs from stored QMAs or new AFs, with conformance to MM-SGQdS, using a QMA transformation called *Composition*.
- *Extending QMA*: to extend a QMA, the designer must search this approach by its name, and then he inerts into the QMA some AF elements (e.g., data model, QoD parameter, QoSPolicy, etc.), which are generated after AF decomposition, to meet RTA requirements. If necessary, new elements can be edited by the expert to be used by the designer.

VII. IMPLEMENTATION

For the framework implementation, we used the Eclipse Meta-modeling Framework (EMF) [4] to benefit from a set of tools, such as reflexive editors or XML serialization of models.

The different meta-models are implemented with *Ecore*, the EMF meta-modeling language. The model transformations are implemented with KerMeta [7]. It is an object oriented meta-modeling language. It does not focus only on structural specifications but supports the specification of the operational semantics of meta-models. Thus, models can be simulated. In addition, it is a powerful model transformation language.

A. Meta-models implementation

MM-SGQdS and the AF meta-models have a tree structure to simplify their reuse through *Decomposition*, *Composition* and *Extension* transformations. An excerpt of MM-SGQdS, displayed with the EMF reflective editor, is illustrated in Figure 6.

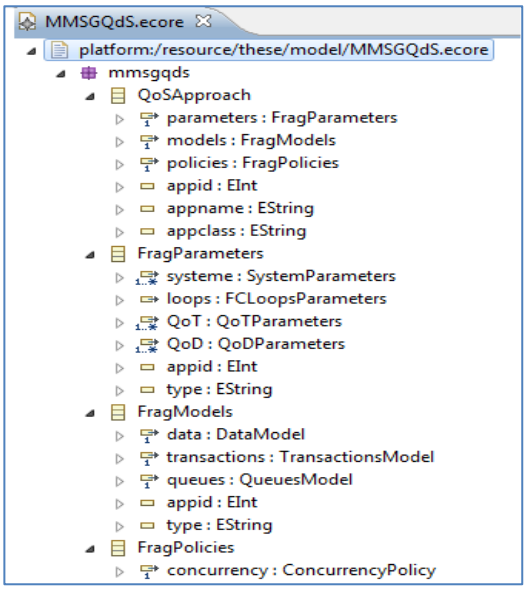


Figure 6. Excerpt of MM-SGQdS tree view.

The meta-class *QoSApproach* is the root of MM-SGQdS. The composition links *parameters*, *models* and *policies* reference respectively the meta-classes *FragParameters*, *FragModels* and *FragPolicies*, the roots of respectively, PA-MM, Mo-MM and PO-MM. The meta-attributes like *appid* and *appname* are QMA properties.

B. Model transformations implementation

A transformation is the automatic generation of a target model from a source model, according to a transformation definition. A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language. A transformation rule is a description of

how one or more constructs in the source language can be transformed into one or more constructs in the target language [25].

In this work, model transformations are applicable to multiple source models and/or multiple target models. We focus on horizontal and exogenous transformations called *Migration* that keep the same level of abstraction (M1) with models expressed using different meta-models [26].

Three categories of model transformations are proposed, namely: *Decomposition*, *Composition* and *Extension*. These transformations are implemented for three model levels: QMA, AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*), and QoS management policies (within the fragment *Frag-Policies*). This results in fifteen KerMeta modules. All these transformations are tested on the FCS-IC1 solution but we cannot present all results in this short paper, neither their implementations.

VIII. EXPERIMENTS

The experiments were carried out on a QMA representing a concrete QMS, called FCS-IC1 [2], which was modeled using a specific QMA editor. We tested the QMA decomposition transformation. The generated AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*) have been used to compose a new QMA, which is identical to the original QMA. We also tested the QMA extension by adding a new QoS Management scenario to the QoS Management Policy within the Frag-Policies fragment of the original QMA.

Other transformations are tested to reuse AF elements, which are very useful for designers. For instance, the *Frag-Models* fragment of the FCS-IC1approach was decomposed into three elements modeling respectively the transaction model, the data model and the queues model. These elements are recomposed to generate a new Frag-Models fragment identically to the original fragment.

A. Expert side experiments

1) Approach edition

The EMF easily generates model editor for an Ecore meta-model to create instances which conform. This allowed us generate QMA editor and editors for each AF. For instance, the QMA editor provides the expert, through a *QoSApproach Editor* menu (Figure 7), a way to instantiate the QMA root meta-class (instance of the *QoSApproach* meta-class) and to configure its related AFs conformant to MM-SGQdS.

The Edited models for QMA or AF modeling are serialized in XMI for their persistence in the database. To display models, EMF provides a graphical mode using its reflexive editor. We developed methods to load AFs from their XMI files for reuse and to display them in a textual mode, which is not provided by EMF.

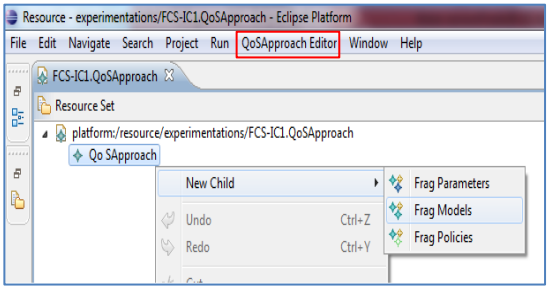


Figure 7. QoS Management Approach Editor.

2) Approach decomposition

To decompose a QMA into AF, we developed a KerMeta module named *decomposeQoSapproaches.kmt*. The method *decompose* of this module parses the QMA from its root.

For each fragment identified through a composition link (instance of *parameters*, *models* or *policies* links in MM-SGQdS) the method instantiates a new fragment conformant to its meta-model (MO-MM, PA-MM or PA-MM). Each AF is serialized in XMI for its persistence.

An excerpt of the graphical display of the Frag-Models fragment, resulting from the FCS-IC1 approach decomposition, is illustrated in Figure 8. This fragment represents the model of the Models component of the FCS-IC1 solution.

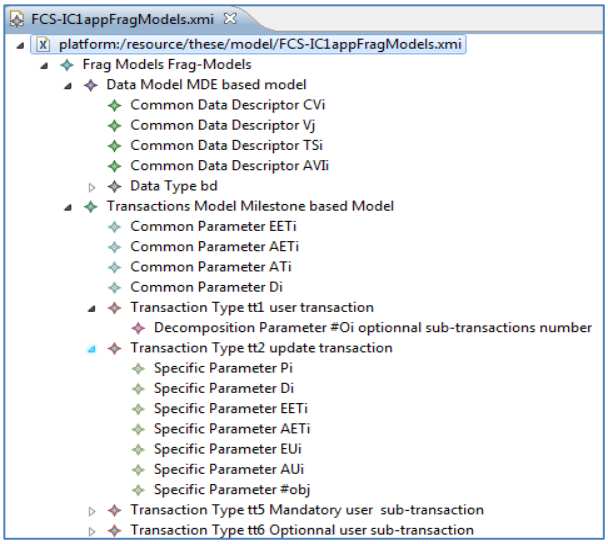


Figure 8. Excerpt of the Frag-Models fragment of FCS-IC1 QMA.

It begins with the configuration of the Frag-Models fragment with conformance to MO-MM. There are detailed the data model elements followed by the transactions model elements.

 ISBN: 978-1-61208-438-1

B. Designer side experiments

1) Approach composition

This transformation generates a new QMA conformant to MM-SGQdS. The new approach is serialized to XMI and displayed in a textual mode. The KerMeta module developed for this purpose is called *composeQoSApproaches.kmt*. It contains a method named *compose* that has a unique parameter, a string array, containing the names of XMI files of serializes AF that will be assembled. For each file, we get the tree structure of the AF by loading its XMI file and meta-model. Then, AF elements are matched to QMA elements and attached to the root of the new QMA. An excerpt of the textual display of the composed FCS-IC1 approach is illustrated in Figure 9. It has the same fragments of the original FCS-IC1 approach.

```
*** Approach : FCS-IC1-composed ***
*** fragment : Frag-Models ***
*Data model: MDE based model
        Principle: Base data are used with imprecision defined by MDE (Maximum Data Error)
        Updating policy : MDE-UpdatePolicy
        Common Descriptors:
                CVi, current value of data di
                Vj, updated value by update transcation j
                TSi, TimeStamp, latest update time of data di
                AVIi, Absolute Validity Interval of data i (AVIi=2*Pj)
        Data Types:
                bd, base data with periodic update
                Specific Descriptors:
                        DEi, data error of data di, [0%, 100%], 100*|CVi-Vj| / |CVi | %
                        AVi, Average value of data di, U(0,100)
                        Vi, value of data di, N : (AVi, AVix×VarFactor), periodic mesure
                        varFactor, data model parameter, U :(0,1), null
*Transaction model: Milestone based Model
        Principle: user transactions are decomposed into one mandatory sub-transaction and many optionnal sub-transaction
        Common Prameters:
                EETi, Estimated Exexution Time of transaction ti
                AETi, Average Exexution Time of transaction ti
```

Figure 9. Textual display of composed FCS-IC1.

2) Approach extension

We tested the extension of the *QoS management policy* of the FCS-IC1 approach (included in its *Frag-Policies* fragment) with a new QoS management scenario. For this purpose, we propose to apply on-demand updates to non-critical data when the system is at maximum overload. In this way, we can avoid the system saturation, during which all coming user transactions are rejected. This new scenario was modeled with corresponding editor (Figure 10) and added to a copy of FCS-IC1approach with conformance to MM-SGQdS.

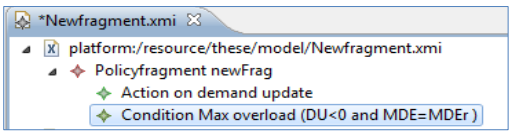


Figure 10. Model of the QoS management scenario used to extend FCS-IC1.

IX. CONCLUSION

In this paper, we proposed an MDE-based framework to automate the reuse of QMA in RTDBMS, which is useful to meet new RTA constraints and QoS requirements. The MDE aims to increase productivity by offering modeling languages tailored to a specific domain, which are typically defined by meta-models. The QMA reuse is based on model transformations, which is the core concept of MDE. Three categories of model transformations are proposed, namely: *Decomposition*, *Composition* and *Extension*. These transformations are implemented for three model levels: QMA, AF (*Frag-Models, Frag-Parameters* and *Frag-Policies*), and QoS management policies (within the fragment *Frag-Policies*). These transformations will be detailed in an extended version of this paper. The framework provides also a methodology consisting of two strategic processes GEN-PM and CONS-PM to help experts and designers achieve their goals with multiple strategies to achieve them, depending on their requirements.

Based on meta-models, we generated editors for QMA and their reusable fragments. We implemented and tested transformations on a concrete QMA. After decomposition, no specificity is ruled out and the rebuilding leads to the initial approach, conformant to the meta-model MM-SGQdS.

Actually, we are interested in providing users a tool for querying the models in order to extract useful knowledge for reuse.

REFERENCES

[1] S. M'barek, L. Baccouche, and H. Ben Ghezala, "An evaluation of QoS management approaches in Real-Time databases". In proc. of the Third International IEEE Conference on Systems. Cancun (ICONS'08), Mexico, 2008, pp. 41-46.

[2] M. Amirijoo, J. Hansson, and S. H. Son, "Specification and Management of QoS in Real-Time Databases Supporting Imprecise Computations", IEEE Transactions on Computers, vol. 55, no. 3, 2006, pp. 304-319.

[3] E. Bouazizi, B. Sadeg, and C. Duvallet, "Improvement of QoD and QoS in RTDBs", In proc. of 14th int. conf. on Real-Time and Network System, Poitiers, France, May 30-31, 2006, pp. 87-95.

[4] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, EMF: Eclipse Modeling Framework, second edition (Eclipse), Addison-Wesley Longman, Amsterdam, 2009.

[5] J. M. Favre, "Towards a Basic Theory to Model Driven Engineering". In Proc. third Workshop in Software Model Engineering (WISME 2004), Lisbon, portugal, 2004, pp. 262-271.

[6] L. C. DiPippo and L. Ma, "A UML package for specifying Real-Time objects", Computer Standards and Interfaces, vol. 22, no. 5, 2000, pp. 307-321.

[7] Z. Drey, C. Faucher, F. Fleurey, and D. Vojtisek, KerMeta language reference manual, 2009.

[8] K. Ramamritham, S. H. Son and L. C. Dipippo, "Real-Time Databases and Data Services", Real-Time Systems, vol. 28, no. 2-3, 2004, pp. 179-215.

[9] Z. Ellouze, N. Louati and R. Bouaziz, "The RTO-RTDB Real-Time Data Model". In Proc. of The 2012 World Congress in

 ISBN: 978-1-61208-438-1

Computer Science, Computer Engineering, and Applied Computing, 2012, pp. 333-339.

[10] J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung, "Imprecise computations", Proceedings of the IEEE, vol. 82, no. 1, 1994, pp. 83-94.

[11] K. D. Kang, S. H. Son and J. A. Stankovic, "Managing Deadline Miss Ratio and Sensor Data Freshness in Real-Time Databases". IEEE Trans. Knowl. Data Eng. vol. 16, no. 10, 2004, pp. 1200-1216..

[12] B. Alaya, C. Duvallet and B. Sadeg, "A new approach to manage QoS in Distributed Multimedia Systems", (IJCSIS) International Journal of Computer Science and Information Security, vol. 2, no. 1, 2009, pp. 1-10.

[13] N. Idoudi , N. Louati, C. Duvallet, R. Bouazizi, B. Sadeg and F. Gargouri, "A Framework to Model Real-Time Databases". International Journal of Computing and Information Sciences (IJCIS), vol. 7, no. 1, 2010, pp. 1–11.

[14] K. Liu, V. C. S Lee., J. K-Y Ng, J. Chen and S. H. Son, "Temporal Data Dissemination in Vehicular Cyber–Physical Systems", IEEE Transactions on intelligent transportation systems, vol. 15, no. 6, 2014, pp. 2419-2431.

[15] N. Louati, C. Duvallet, R. Bouaziz, and B. Sadeg, "RTO-RTDB: A Real-Time object-oriented database model", In proc. of the 23rd IASTED International Conference on Parallel and Distributed Computing and Systems, Dallas, USA, 2011, pp. 1-9.

[16] C. Lu, J. A. Stankovic, G. Tao and S.H. Son, "Feedback control Real-Time scheduling: Framework, modeling and algorithms", In Journal of Real-Time Systems, vol. 23, no. 1, 2002, pp.85-126.

[17] OMG. Meta Object Facility (MOF) specification - version 1.4, formal/01-11-02, Avril 2002.

[18] K. Ramamritham, S. H. Son and L. C. Dipippo, "Real-Time Databases and Data Services", Real-Time Systems, vol. 28, no 2-3, 2004, pp. 179-215.

[19] R. Deneckere, E. Kornyshova and C. Rolland, "Enhancing the Guidance of the Intentional Model MAP: Graph Theory Application", IEEE 3rd int. conf. on Research Challenges in Information Science RCIS'09, Fes, Morocco, 2009, pp. 13-22.

[20] J. A. Stankovic and S. H. Son, "Architecture and Object Model for Distributed Object-Oriented Real-Time Databases", In Proc. 1st Int. Symp. on Object Oriented Real-Time Distributed Computing, Kyoto, Japan, 1998, pp. 414–424..

[21] T. Niklander and K. Raatikainen, "RODAIN: A Highly Available Real-Time MainMemory Database System". In Proc. of the IEEE International Computer Performance and Dependability Symposium, Durham, NC, 1998, pp. 271-.

[22] B. Alaya, C. Duvallet and B. Sadeg, "Feedback architecture for multimedia systems". In proc. of the IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2010), 2010, pp. 16-19.

[23] S. Hamdi, E. Bouazizi and S. Faiz, "A new QoS Management Approach in Real-Time GIS with heterogeneous Real-Time geospatial data using a feedback control scheduling". In Proc. 19th International Database Engineering & Applications Symposium (IDEAS'15), 2015, pp.174-179

[24] C. Atkinson, and T. Kühne, "Model-driven development: A metamodeling foundation". IEEE Software, vol. 20, no. 5, 2003, pp. 36–41.

[25] A. Kleppe, J. Warmer and W. Bast., MDA Explained, The Model-Driven Architecture: Practice and Promise. Addison Wesley. 2003.

[26] T. Mens, P. Van Gorp, "A taxonomy of model transformation". Electronic Notes in Theoretical Computer Science 152, 2006, pp. 125–142.

 ISBN: 978-1-61208-438-1

An Approach for Reusing Software Process Elements based on Reusable Asset Specification: a Software Product Line Case Study

Karen D. R. Pacini
and Rosana T. V. Braga

Institute of Mathematics and Computer Sciences
University of São Paulo
São Carlos, SP, Brazil
Email: karenr@icmc.usp.br, rtvb@icmc.usp.br

***Abstract*—Software reuse is becoming an important focus of both academic and industrial research since the rising demand for new software products and technologies is constantly growing. The short time to market, limited resources and lack of specialists are the main reasons for this investment on software reuse. As long as customers demand speed to deliver, there is a increasing special concern about software quality. In this context, we propose an approach to support both better time to market and software quality from reusing software process elements using the Reusable Asset Specification (RAS). This approach presents a mapping structure to represent process elements as reusable assets. The sharing of process elements among several projects aims to decrease time spent on defining the process model, as well as reducing the space used to store processes and their elements. Documenting these processes will also be facilitated, since it is possible to reuse a whole process or process's sub-trees that have already been documented or even certified. To illustrate our approach, we present a case study where a Software Product Line (SPL) process is mapped to RAS, highlighting the issues raised during the mapping and how we proposed to solve them.**

Keywords–Software Process; Process Reuse; Software Product Line; RAS; Reusable Asset Specification.

I. INTRODUCTION

The software industry has been adapting to the large increase of demand arising from the constant evolution of technology. The concept of software reuse gets an important role on this new way of software manufacturing, in which development time is reduced, while quality is improved [1]. Software product lines (SPL) emerged in this context, to support reuse by building systems tailored specifically for the needs of particular customers or groups of customers [1]. Reuse in SPL is systematic – it is planned and executed for each artifact resulting from the development process.

The most common SPL development approaches, such as Product Line UML-Based Software Engineering (PLUS) [2], Product Line Practice (PLP) [1], etc., are focused on the process to support the domain engineering and/or the application engineering, without considering the computational tools that support the process. Indeed, the choice and use of tools are made apart from the process and are strongly associated to variability management, i.e., dealing with the definition of the feature model and its mapping to the artifacts that implement each feature. Some examples of these tools include Pure::Variants [3], Gears [4], and GenArch [5].

To support a uniform representation of reusable assets, in 2005 the Object Management Group (OMG) has proposed the Reusable Asset Specification (RAS), which allows a common approach to be used by developers when storing reusable assets [6]. RAS offers a basic structure (CORE), but allows the creation of extension modules in order to adequate to the particular needs of each project. The specification is available via XML Schema Definition (XSD) and XML Metadata Interchange (XMI) files and its usage is defined by profiles. In the particular case of SPL, the use of RAS to model repositories contributes to make assets compatible to each other.

Several extensions to the original RAS profile have been presented, however their focus is on improving the representation of specific types of reusable assets [7][8][9]. However, we have not found any works showing how RAS could be used to represent the elements of a process, in particular in the SPL domain. This is not a trivial task, as there are several decisions to be made, for example, how process elements that are compositions of other elements should be stored using RAS. This also motivated this work, as we are interested in extending reuse to the process level, i.e., to facilitate the reuse of process phases, activities, or any other assets related to the process itself. In the particular case of SPL, it is important to consider approaches successfully applied in practice and well documented, such as the approaches proposed by Gomaa [2] and by Clements [1], for creating the case study to apply our approach. The use of these approaches is supported by their wide documentation and can illustrate scenarios for a variety of real applications.

So, the main motivation for this work is that, although SPL development approaches focus on establishing the process itself, only SPL artifacts are considered as reusable assets, rather than the process elements that could bring a number of benefits if appropriately reused. Indeed, current approaches do not motivate process reuse, which causes rework each time a process needs to be instantiated from the general processes. Additionally, experience gained from successful projects executed in the past are not taken into account when new similar projects arise, because they were not adequately stored for reuse.

Therefore, considering this context, this paper aims at proposing an approach that allows the storage of process elements using RAS, in particular in the SPL domain. This

 ISBN: 978-1-61208-438-1

can leverage the reuse of each SPL process element across several projects, in an independent way, potentially increasing reuse. As process elements can be composed of other process elements, reuse can be done both for single elements or elements in higher levels of the hierarchy, which contain one or more elements. So, the paper has also the objective of describing the main problems found when trying to map process elements to RAS and gives insights on how this has been solved by our approach.

Software & Systems Process Engineering Meta-model (SPEM) version 2.0 [10] was the process meta-model that inspired our approach. This is because SPEM is being widely employed as a software process modeling language, as indicated by Garcia-Borgonon [11]. It was used as basis to define our own structure, which represents several types of processes, as presented in the paper. It is important to notice that, although this paper presents a SPL process to exemplify the approach, any software processes that matches the model proposed in this paper could be used.

The remainder of this paper is organized as follows. Section II presents some background on SPL development and RAS. Section III presents our approach to store SPL process elements as reusable assets using RAS. Section IV presents a case study to illustrate our approach. Section V discusses related work and, finally, Section VI presents conclusions and future work.

II. BACKGROUND

In this section, we describe a SPL technique (PLUS) and a reuse standard (RAS) that have been used as a foundation for the proposed approach, so they are important to allow its understanding.

A. SPL Development Process: the PLUS approach

PLUS [2] is the SPL process chosen as a case study to illustrate the approach presented in this paper. However, any other SPL process could have been used, since the main idea is to illustrate how a SPL process can be represented using RAS.

PLUS employs methods based on the Unified Modeling Language (UML) [12] to develop and manage SPLs. Its main goal is to model features and variabilities of an SPL. The approach is based on the Rational Unified Process (RUP) [13] and each phase corresponds to a RUP work-flow with the same name.

The process used by PLUS is evolutionary and has two main activities: the product line engineering (or Domain Engineering) and the Application Engineering (configuration of the target system that results in a new product). For each activity, either in Domain or Application Engineering, there is a corresponding evolutionary process (Evolutionary Software Product Line Engineering Process - ESPLEP).

According to Gomaa [2], ESPLEP life cycle for Domain Engineering is composed of five activities with the three most important: requirements modeling, analysis modeling and design modeling, as can be observed in Figure 1. During requirements modeling, the SPL scope is defined, resulting in use cases and feature models. The analysis modeling includes static modeling, dynamic modeling, finite state machine modeling, as well as the construction of objects and analysis of dependencies between features and/or classes. The design modeling involves the definition of the SPL architecture.

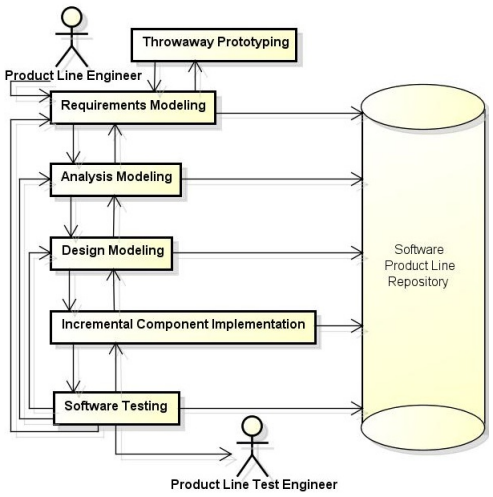


Figure 1. Process for applying the PLUS approach to SPL Engineering Phase - Adapted from Gomaa [2]

These are the basic activities, but variations can be added to the process. A characteristic of this approach is that stereotypes are used in the diagrams to identify different types of use cases, class diagrams or object diagrams (e.g., to denote mandatory or optional features).

B. Reusable Asset Specification

There is an increasing demand to ease software reuse, as it involves high costs associated to creating, searching, understanding, and using the assets found in a specific context. So, the creation of standards to organize and package assets is necessary. In this context, the OMG has proposed the Reusable Assets Specification (RAS) [6], which is a group of object management standards to allow the packaging of digital assets to improve their reusability.

RAS supplies, through a consistent standard, a set of guidelines that help to structure reusable assets. This can reduce conflicts that would arise when trying to reuse them. RAS models are based on UML [12] and Extensible Markup Language (XML).

The specification describes the reusable assets based on a model called Core RAS, which represent the fundamental elements of an asset. This core model can be adapted if necessary, by creating Profiles. RAS Profiles are a formal extension of core structure, which allows to add or improve information according to a specific context. They can be created to introduce more rigid semantics or constraints, however they must not change core definition or semantics. OMG supplies the default profile, based only in Core RAS, and also two customized profiles to be used in specific situations: the default component profile to support the principles and concepts of software components, and the default Web Service Profile that describes the client portion of a web service. Other extensions exist as well [7].

Core RAS packaging structure is split in five major sections (or entities): Classification, Usage, Solution, Related Assets and Profile, as shown in Figure 2.

 ISBN: 978-1-61208-438-1

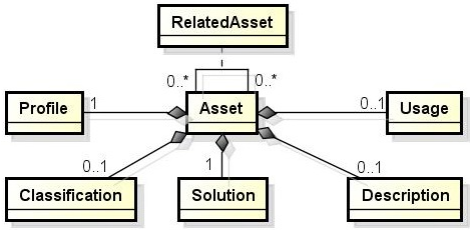


Figure 2. Core RAS Major Sections, Adapted from OMG [6]

Classification is a container entity used to allow the classification of assets to ease their further retrieval. It can contain descriptors, tags, and values, besides the context (domain, development, test, deployment, etc.). Usage contains usage instructions that improve the understanding of the asset before its usage, as well as how to perform the customization of the variation points. Solution contains the artifacts of the asset, which can be requirements, models, code, tests, documents, among others; Related Assets - describes other assets related to this one, together with the relationship type (e.g., aggregated, similar, dependent). Profile defines the version of the profile to which the asset refers to (e.g., the default profile, or any other extensions).

III. The Proposal

Although different SPL development processes share a common basis, they also contain variabilities. This motivates representing them as reusable assets. Therefore, in this paper, we propose to store process elements regarding SPL development as reusable assets using RAS (see Section II-B). In order to accomplish that, it has been defined a process modeling structure to represent processes.

A. Process Modeling Structure

The process modeling structure used in our approach is shown in Figure 3. It contains several elements: process model, phase, activity, artifact, etc. This structure was inspired on OMG SPEM 2.0 [10] concepts, and aims to represent all process elements for a software development process, i.e., it must be capable of representing processes from different development approaches existing in the literature (in the SPL domain we can mention ESPLEP - see Section II-A).

It is important to notice that this model is used to represent both the process template (i.e., the process model as defined by its authors) and the process instance, which is derived by instantiating the process template for particular purposes. A process instance refers to a template but has its own elements, according to the process execution. This is important because we may want to reuse not only the templates, but also the instances that were successful in a particular context and thus can be recommended when similar situations occur. For example, PLUS is a process model (template) that can be reused in a concrete SPL project, resulting in a process instance. Later, when a new project begins in a similar context, instead of reusing PLUS, we might want to reuse the instance instead, because it is already customized to the new context.

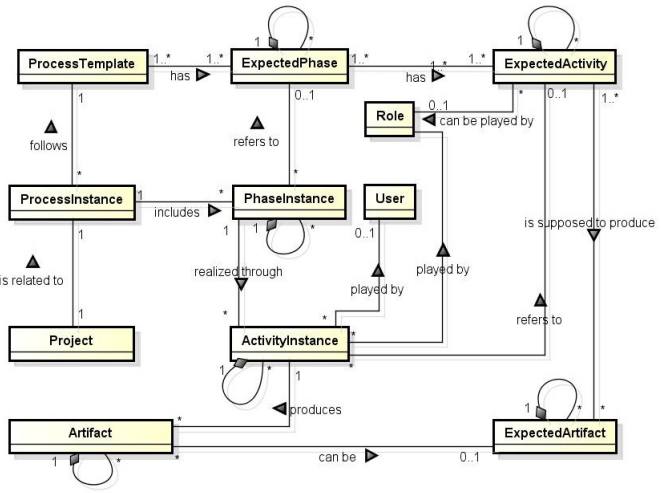


Figure 3. Process Model derived from SPEM

B. Modeling Structure Applied to Process Template and Instance

To illustrate the usage of the model proposed in Figure 3 we show, in Figure 4, the example of a random software development enterprise that is developing a SPL to ease the development of applications for hotel management (this example was chosen as the business is simple and it helps to understand the difference between process template and process instance). The development process was instantiated from ESPLEP (Figure 1) and here we focus only on the domain engineering phase. The domain engineer starts by creating a specific Project (*Hotel SPL Project*) and, associated to it, a Process instance (*Hotel SPL Process*), which in turn is associated to ESPLEP. In the figure, we use stereotypes to identify the roles played by each class in the example and [...] to express that other analogous instantiations can take place.

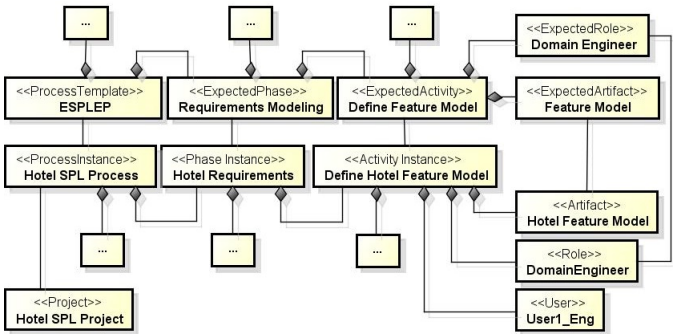


Figure 4. Example of ESPLEP instantiation on the proposed Process Model.

For each ESPLEP process element, we have to analyse whether it is adequate to our Hotel SPL process and, if so, create the respective instance. Additionally, the Hotel SPL process can be adapted to the particular context of the enterprise, for example adding new activities or artifacts, skipping optional activities, etc., as long as ESPLEP allows these adaptations. This is possible because, during the definition of ESPLEP modeling, it has been defined which elements are mandatory or optional. Mandatory elements need to have an instance, while

 ISBN: 978-1-61208-438-1

optional elements can be omitted.

C. Mapping Process Elements to RAS

We propose to map each process element to an independent RAS *Asset*, as shown in Figure 5. We discuss the rationale for that later in Section III-D. In the figure, this mapping is done considering the example of a template process and its associated elements. At the left-hand side of the figure, we have the process template, in the middle we have RAS, and at the right-hand side, examples of values assumed by the properties.

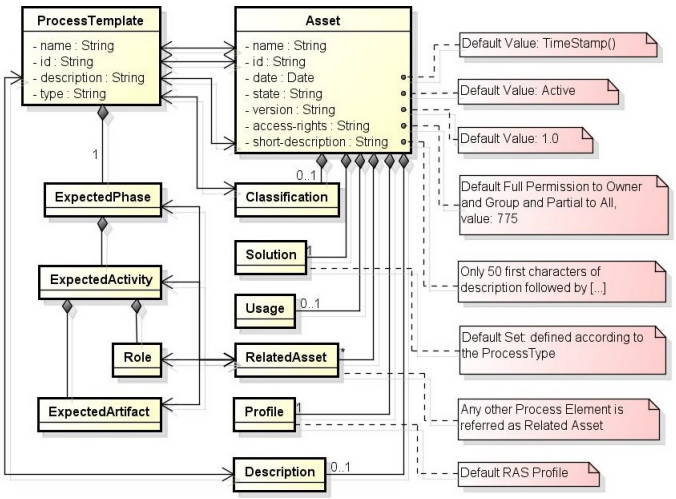


Figure 5. Example of a Mapping Structure from Process Template to RAS.

As presented in the figure, the attributes *name* and *id* have a direct relationship in both structures (*Process Template* and *Asset*). The *description* attribute is mapped to the *Description* RAS attribute, but is limited to the first 50 characters of the *short-description* attribute from the RAS *Asset*. The *type* attribute is mapped to the *Classification* element in RAS. The other process elements are treated as independent *Assets*, so they are created separately in the RAS structure and then related to each other through the *RelatedAsset* RAS element.

After doing this mapping, all the process information is stored in a RAS structure, however, according to RAS, some mandatory elements still need to be filled in. For example, the *Asset* has attributes: *date*, *state*, *version* and *access-rights*, which can be completed with default values as shown in Figure 5. The *Profile* element is mandatory and identifies which profile is being used to represent the *Asset*. In this case, we are using the *DefaultProfile*, version 2.1, as shown later in Section IV-A.

Another important mandatory element in the RAS structure is the *Solution*, which has to be filled in with information and related documents corresponding to the process element being stored. For example, in the Process Template the representation could be a file containing the complete structure of the process in an XML document. There are other optional elements exemplified in the case study.

D. Increasing the Potential Reuse

An important goal of the proposed approach is to enhance the potential reuse of processes, as well as decreasing time, effort, and storage space when creating and instantiating processes. By storing each element independently as a reusable asset, it is possible to share it among different processes and process instances, as illustrated in Figure 6. When an element is shared among processes, the whole tree with related elements associated to the root element is shared as well.

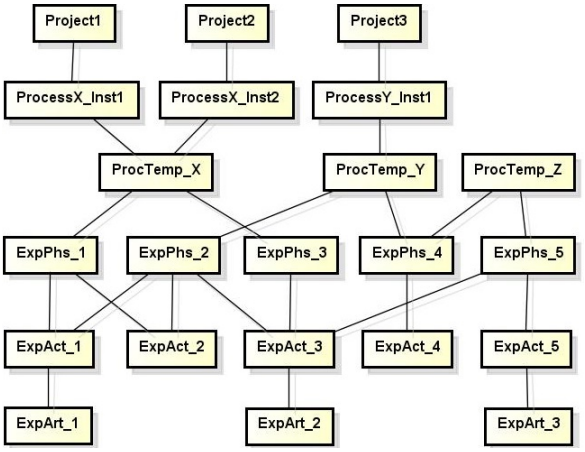


Figure 6. Example of Potential Reuse

As can be observed in Figure 6, the process element identified by ExpAct_3 represents an expected process activity, which is being referenced by three different expected phases. These phases are referenced by two different processes. The reuse of process elements as suggested by our approach allows we to share elements both in the process template and in the process instances, although the example in Figure 6 illustrates reuse only in template elements. This leads to a greater reuse potential, besides the fact that any repository or tool based on RAS can be easily used.

IV. Case Study

As described in the previous section, our approach allows the representation of process elements according to RAS, both for the process template and its execution. ESPLEP is composed of five expected phases (see Section II-A): Requirements Modeling, Analysis Modeling, Design Modeling, Components Incremental Implementation, and Software Testing. Each phase is composed of expected activities and their expected artifacts.

RAS is flexible regarding the possible ways to use, extend, and represent assets according to each project needs. Our approach is one of many possible ways to use it. We recommend to follow this usage pattern to ease the retrieval of assets later, i.e., client applications for searching assets will be easier to implement if they know that the underlying structure is based on RAS. However, it is important to observe that there is a minimum set of information required for a reusable asset to be considered in conformance with RAS: it has to indicate the used profile, at least one artifact and the basic information about the asset, as shown in Figure 5. Additionally, there are other information that can also be stored, so this case study aims at showing a possible way of using RAS elements to store process elements. This is shown in the following subsections.

A. Profile

The *Profile* element (Figure 7) refers to the asset representation structure that is being currently used. The RAS (*Core*)

 ISBN: 978-1-61208-438-1

is abstract, thus only profiles are instantiated and all of them are derived from the core. The derived profile that is closest to the core, and was chosen to be used in this work, is called *Default Profile*. More specifically, we adopted version 2.1, as it is compatible to any profile extending it.

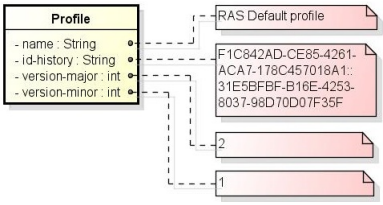


Figure 7. Example of use of Profile RAS Element.

B. Solution

The *Solution* element refers to the artifacts that compose the reusable asset. An asset can be seen as a set of artifacts (at least one artifact is required). The artifacts are the main reuse goal, and they can be classified into several types, like documents (doc, pdf, txt), code (java, sql, php, C#), descriptors (XML, XSD, HTML), and others.

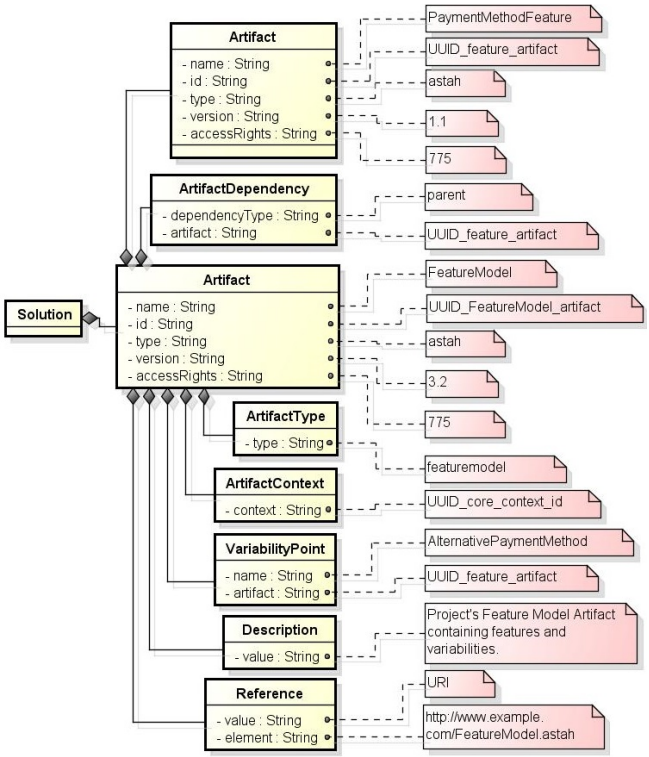


Figure 8. Example of use of Solution RAS Element.

Figure 8 presents an instance of the *Solution* element. Considering the hotel SPL introduced on Section III-B, the solution contains the Feature Model, defined during the *Define Hotel Feature Model* activity of the *Hotel Requirements* phase.

The main artifact stores (in fact it is a reference to where the real object is) the model itself, and has attributes such as name, identifier, type (file type, e.g., .astah), version, and access rights of the artifact, which we have defined as 775, following the Unix model, i.e., the artifact owner and the group to which he belongs to have total rights, while other users can only read and execute. RAS also recommends the use of Universally Unique IDentifiers (UUIDs).

The *ArtifactType* represents the artifact logical type, indicating what the artifact represents in the model (in the example, it is a SPL Feature Model). The *ArtifactContext* represents the context in which this artifact is useful. In the example, as the feature model is essential for the asset, it is classified with the *Core* type, as shown in the figure.

The *VariabilityPoint* element describes artifact variabilities. In the example, the artifact has a feature called PaymentMethod, which has several different alternatives. In this case, the feature that has variability is presented in this element, while the corresponding alternative features and variability rules are defined in the *Usage* element described later. Also, this artifact has other dependent artifacts representing each feature of the feature model. This allows the reuse of the feature model itself and the corresponding artifacts that implement them, not only during application engineering, but also in the domain engineering of other SPLS of the same domain (for example, PaymentMethod could be used in many different SPLs).

C. Classification

The *Classification* element refers to the asset descriptors or classifiers. It can include more than one descriptor or even schemas to describe the asset. It is also possible to define the contexts that will be referenced by artifacts and by the *Usage* and its activities.

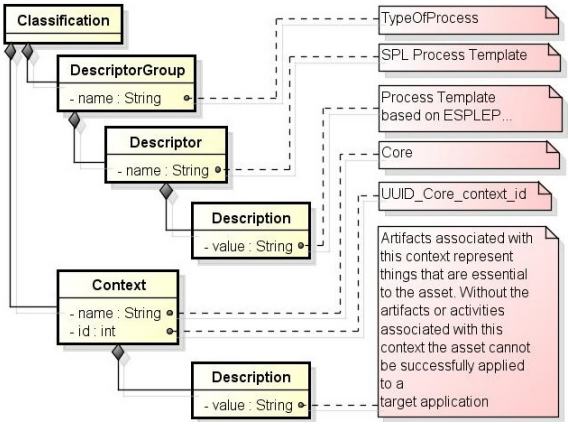


Figure 9. Example of use of Classification RAS Element.

Figure 9 presents an instance of the *Classification* element. Assume that we want to classify the *SPL Process Template* process element. So, the "SPL Process Template" is a *Type of Process* kind of classification. To map this information into RAS, first we need to define a group of classifiers that represent what kind of classification we are making, in this case, we are classifying as *TypeOfProcess*, so this will be the name of our *DescriptorGroup* RAS element. Defined that, we can define the value for this type, each value is defined as an instance of *Descriptor* RAS element, which in this case is *SPL Process Template*. An asset may have many classifiers, for example this same asset could be classified by DescriptorGroup *ProcessElement* and Descriptor *ProcessTemplate*.

 ISBN: 978-1-61208-438-1

In an analogous way, all the other process elements can be classified using this structure, for instance *SPL Process Instance* which is also part of *TypeOfProcess* DescriptorGroup; *Expected Phase*, *Expected Phase* and *Expected Artifact*, which are also part of *ProcessElement* DescriptorGroup; and so on.

The *Context* element is used to refer to artifacts and activities in the Asset context. As shown in the figure, the *Core* context represent that the related artifact/activity is essential to the asset.

D. Usage

The *Usage* element is used to keep information about how to use the asset (manuals for example), as well as which tasks have to be executed in order to that asset works correctly. This information can be relative to the context, to a specific artifact, or to the whole asset.

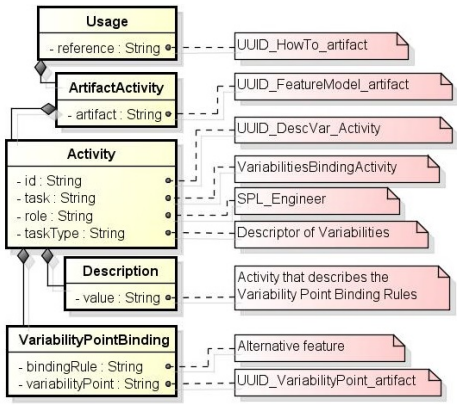


Figure 10. Example of use of Usage RAS Element.

Figure 10 presents an instance of the *Usage* element, in which the *reference* attribute refers to an artifact contained in another asset, identified by *UUID_HowTo_artifact*. This artifact represents a document with instructions on how to use and interpret the components of the *Usage* element. Even though this artifact is not mandatory, it can be useful to improve reuse. Schemas and other types of artifacts can also be referenced.

In the hotel SPL example, as mentioned before, there is a feature named PaymentMethod that represents a variability. Figure 10 presents the rules for using this variability from instances of the *VariabilityPointBinding* element. These instances are contained in an *Activity* element, which itself is contained in an *ArtifactActivity* element. This means that the activity is relevant only in the context of the referenced artifact. The activity (*VariabilitiesBindingActivity*) describes the rules to be followed for binding variabilities of the Feature Model artifact.

In the example, the artifact identified by *UUID_feature_artifact* has a variability called *AlternativePaymentMethods*. According to the rule defined in *VariabilityPointBinding*, the dependent artifacts (children) refer to alternative features of the Feature Model as defined in the *bindingRule* attribute.

E. Related Assets

The *Related Asset* element specifies the relationships among reusable assets. From these relationships, it is possible to assemble a dependency tree with all related assets.

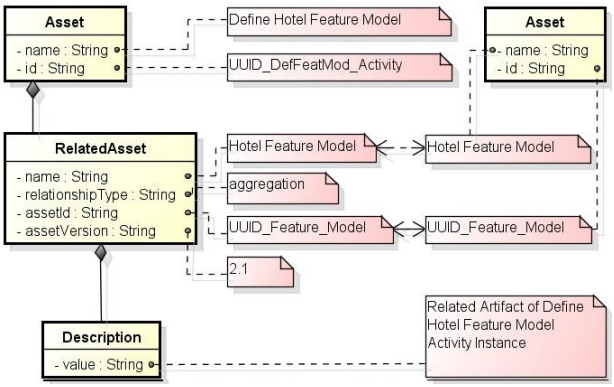


Figure 11. Example of use of Related Asset RAS Element.

Figure 11 presents an instance of *Related Asset*. It has a *name* attribute that corresponds to the name of the asset being related with, as well as *assetID* and and *assetVersion* representing the ID and version of the related asset, respectively. RAS defines some types of relationship (e.g., an *aggregation* in the figure) and allows other types to be created.

In this example, there is a relationship between <<ActivityInstance>> Define Hotel Feature Model and <<Artifact>> Hotel Feature Model. In this scenario, only the activity is related to the artifact, not the opposite way. This means that only the activity has visibility of the artifact. If the visibility was supposed to be both ways the artifact should have a relationship with the activity as *parent* type.

Navegability on relationships is very important on the reuse context. For example, when selecting an element for reuse, all the dependencies of this element will be loaded as well. In this case, if someone wishes to reuse the activity of Figure 11, the artifact would be loaded with it. But if they want to reuse just the artifact, it is possible because the artifact has no relationships (dependencies). Thus, if an element depends on another element and they must be loaded together, the relationship must be defined in both elements.

V. Related Work

While searching for RAS related studies on the literature, we could not find any descriptions or examples of how they use RAS to represent and to pack their reusable assets. Most of the studies focus on presenting how to identify and use the assets, rather than on how to map them into the RAS structure.

Part of the studies found on our research proposes RAS extensions to fit to several different purposes. The application of RAS in these studies are usually very specific to each case, for example to store components, or services, or process generated artifacts and so on. One of proposed extensions is presented by Mikyeong Moon et al. [8]. They propose an extension of the RAS Default Profile to store, manage, and trace variabilities on a Software Product Line.

Another example of using RAS to represent reusable assets is proposed by Islam Elgedawy et al. [14]. They propose to use the specification to represent Component Business Maps (CBMs) to allow the early identification of reusable assets in a project. They do not specify which RAS Profile they use or if they created their own extension to represent their assets.

 ISBN: 978-1-61208-438-1

An example of RAS application that does not use new extensions is proposed by Nianjun Zhou et al. [15]. In their approach, they present a legacy reuse analysis and integration method to support modeling legacy assets in a SOA context. To store the assets extracted by their approach they use the IBM Rational Asset Manager Repository (RAM), which is typically used for storage of unstructured assets (jar, war and ear) and documents specified using RAS.

There are other online repositories based on RAS in the web, one example is LAVOI created by Moura [7] and OpenCom created by Ren Hong-min1 et al. [9]. Both extended the RAS profile to adapt it to a wide range of types of assets and to facilitate assets classification, search and use.

Although there is a number of works related to RAS, none of them brings explicit examples of how to use RAS and to map the attributes as this work does. In addition, no studies were found that suggest process elements reuse based on RAS.

VI. Conclusions and Future Work

This work presented an approach to represent process elements as reusable assets using the RAS. For this, a mapping of these elements into the RAS structure was presented. The mapping not only represents the main information of process elements into RAS mandatory structure but also guides the user on how to use several other structures that are available at the RAS Default Profile. This representation makes possible the creation of a repository of process elements, which may highly increase the potential of reuse. Reusing processes and process elements has lots of benefits, such as improving time to market, decreasing time spent and staff effort, increasing quality. Besides that, a repository may be built with mechanisms to recommend process and process elements according to the user type and application context.

The contribution of this paper is applicable not only in the SPL context, but in any software processes in other contexts, as long as they follow our proposed meta-model structure derived from SPEM. For processes with different structures, a mapping analogous to that provided here can be done.

Another advantage of using the proposed approach is that process elements can also be shared among different project contexts, both for template and instance applications. An additional contribution of this paper is to serve as a documented example of how to use RAS in a practical way, since no example of usage details was found on our research in the literature.

As future work, we will implement a Service Based Tool for representing process elements into RAS Structure. This tool will be able to get input parameters relative to each element information (attributes) and generate its RAS mapping to store them into any repository that can read RAS files as input. In addition to process elements, this tool will be able to map any reusable artifact generated from the development process to RAS Structure. Thus, this tool will provide to the users, services to support the management of assets of SPL development, from process to maintenance.

Acknowledgements

Our thanks to Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and University of São Paulo (USP) for financial support.

References

[1] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns. Addison Wesley Professional, 2002, the SEI series in software engineering.

[2] H. Gomaa, “Designing software product lines with uml 2.0: From use cases to pattern-based software architectures,” in Reuse of Off-the-Shelf Components. Springer, 2006, pp. 440–440.

[3] D. Beuche, “Modeling and building software product lines with pure::variants,” in 16th International Software Product Line Conference-Volume 2. ACM, 2012, pp. 255–255.

[4] R. Flores, C. Krueger, and P. Clements, “Mega-scale product line engineering at general motors,” in Proceedings of the 16th International Software Product Line Conference-Volume 1. ACM, 2012, pp. 259–268.

[5] E. Cirilo, U. Kulesza, and C. J. P. de Lucena, “A product derivation tool based on model-driven techniques and annotations.” Journal of Universal Computer Science (JUCS), vol. 14, no. 8, 2008, pp. 1344–1367.

[6] O. M. Group, “Reusable asset specification,” OMG, 2005. [Online]. Available: http://www.omg.org/spec/RAS/2.2/ [Retrieved: Sep, 2015]

[7] D. d. S. Moura, “Software profile ras: extending ras and building an asset repository,” Master’s thesis, 2013. [Online]. Available: http://www.lume.ufrgs.br/handle/10183/87582

[8] M. Moon, H. S. Chae, T. Nam, and K. Yeom, “A metamodeling approach to tracing variability between requirements and architecture in software product lines,” in 7th IEEE International Conference on Computer and Information Technology (CIT). IEEE, 2007, pp. 927–933.

[9] R. Hong-min, Y. Zhi-ying, and Z. Jing-zhou, “Design and implementation of ras-based open source software repository,” in 6th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 2. IEEE, 2009, pp. 219–223.

[10] O. M. Group, “Software & systems process engineering metamodel specification,” OMG, 2008. [Online]. Available: http://www.omg.org/spec/SPEM/2.0/ [Retrieved: Sep, 2015]

[11] L. García-Borgoñón, M. A. Barcelona, J. A. García-García, M. Alba, and M. J. Escalona, “Software process modeling languages: A systematic literature review,” Inf. Softw. Technol., vol. 56, no. 2, Feb. 2014, pp. 103–116.

[12] O. M. Group., “Unified modeling language,” OMG, 2011. [Online]. Available: http://www.omg.org/spec/UML/2.4.1/ [Retrieved: Sep, 2015]

[13] R. S. Corporation, “Rational unified process,” IBM, 1998. [Online]. Available: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf [Retrieved: Sep, 2015]

[14] I. Elgedawy and L. Ramaswamy, “Rapid identification approach for reusable soa assets using component business maps,” in IEEE International Conference on Web Services (ICWS). IEEE, 2009, pp. 599–606.

[15] N. Zhou, L.-J. Zhang, Y.-M. Chee, and L. Chen, “Legacy asset analysis and integration in model-driven soa solution,” in IEEE International Conference on Services Computing (SCC). IEEE, 2010, pp. 554–561.

 ISBN: 978-1-61208-438-1

An Extensible Platform for the Treatment of Heterogeneous Data in Smart Cities

Cícero Alves da Silva and Gibeon Soares de Aquino Júnior

Department of Informatics and Applied Mathematics
Federal University of Rio Grande do Norte
Natal, RN, Brazil
Email: cicerojprn@gmail.com, gibeon@dimap.ufrn.br

***Abstract*—Nowadays, there is a lot of devices of varying technologies in the urban environment, which makes the integration of data generated by them a difficult process due to their heterogeneity. However, it is important to manage these data in an integrated way to enable the exchange of information between existing fields and assisting in the decision-making process. Moreover, there is no way to tell how these data will need to be processed since each application may require it to be available obeying specific processes. Thus, this article describes the design and implementation of a platform that aims to integrate, process and make available data streams from heterogeneous sources. It also defines an extensible data processing flow, which makes the creation of new processes for existing data and the inclusion of new types of data easier. Finally, a case study was conducted, which used a parking lot as scenario and assessed extensibility and performance aspects related to platform implementation.**

Keywords–Smart Cities; Software Architecture; Extensibility.

I. INTRODUCTION

The widespread use of intelligent devices and other types of sensors resulted in the emergence of the Internet of Things (IoT), a paradigm in which the objects of the everyday life are equipped and able to communicate with other objects and users, which makes them a part of the Internet [1]. Thus, these objects are able to work in different urban environments, providing data that are collected in them and enabling the Smart Cities concept to be used. Even though the term "Smart City" has been widely used nowadays, it does not present a standardization regarding its meaning. However, it is known that a smart city should pay special attention to performance improvement in six different areas: Economy, People, Governance, Mobility, Environment and Living [2][3].

Nonetheless, only the use of these objects is not enough to improve urban life [4]. It is important that the management of the data generated in them is carried out in the same place, allowing the exchange of information between the existing sectors to happen and assisting in the decision-making process. However, this data integration is not a trivial task because of the devices' heterogeneity [1][5][6], since they use different technologies and different communication protocols and produce data flows with multiple formats and different characteristics.

Furthermore, applications that consume these data may require them to be made available in different forms, making it necessary for them to be processes before its delivery. Therefore, to assist in the comprehension of the complied problems, the systems that propose to process data flows from these heterogeneous sources need to filter them, combine them and assemble them, thus producing new data as output [7]. However, there is no telling in which form the data needs to be processed, since the same data may need to be processed in different ways to meet the application's needs and since there may also be the need to perform the inclusion of new types of data in the platform.

Finally, this article discusses the definition, design and implementation of a Smart City platform whose focus is related to the integration, prossessing and availability of data flows from heterogeneous sources in an urban environment. In addition, this study also discusses the process of extensible data processing defined in this platform, which allows the data to be processed according to its specific characteristics and the application's needs. Section II discusses a few related works. Subsequently, Section III shows the platform proposed in this study. Section IV, in turn, discusses a case study that used a parking lot scenario and assessed some important aspects related to the implementation of the platform. Finally, Section V shows this study's conclusions and future work.

II. RELATED WORK

In the study of Anthopoulos and Fitsilis [8], a research is held in smart cities in order to develop an architecture to be used in the management of urban services. However, unlike the present study, Anthopoulos and Fitsilis' work deals only with the architecture's description. Thus, it is not possible to identify the modules that must be implemented for the proposed layers to work and it is not possible to assure that these layers are effective to work with the data generated in the urban environment.

In Filipponi et al.'s work [9], an event-based architecture that allows the management of heterogeneous sensors to monitor public spaces is presented. However, this architecture is different from the one proposed in this work, since its use is very limited and it does not incorporate many requirements such as privacy and monetization.

The MAGIC Broker 2 platform, which focuses on objects' interoperability and proposes to work in an IoT environment, is presented in Blackstock et al.'s article [10]. However, the authors state that this platform is not ready to work in a Smart City environment. In contrast, in the present work, the platform is designed precisely to deal with this area of study.

 ISBN: 978-1-61208-438-1

Middlewares for IoT are proposed in Gama, Touseau and Donsez's study [6] and in Valente and Martins' study [11]. However, they differ from the platform proposed in this study since they do not perform the extraction of knowledge from the integrated data and do not have privacy strategies for the transfered information.

Andreini et al.'s article [12] discusses an architecture based on the principles of service orientation. However, it is limited to smart objects' geographical location issues. Furthermore, it does not address data privacy and does not allow the aggregation and extraction of the knowledge found in them.

III. PROPOSED PLATFORM

The platform proposed in this paper aims to enable integration, processing and availability of different types of data generated by the sensors that exist in the urban environment. Furthermore, it focuses on providing the extensibility of processing tasks performed on these data due to the fact that it is not possible to predict in what form their flows need to be processed.

Thus, the extensibility of the processing steps is important due to the fact that the applications are so dynamic and may require different processing ways for the same data flow and due to the fact that with time, new types of data will turn up as a result of the emergence of a new source.

This way, moving from the intended goals to the platform and seeking to provide this extensible data processing feature, the following requirements were defined:

1) Data retrieval from sources with heterogeneous technologies;
2) Availability of data integrated into the platform;
3) Data association allowing information from different domains to be combined to work in an unified manner;
4) Follow the modular and "pluggable" approaches, making the maintenance and extension of the platform easier;
5) Have a well defined data transformation process, since its modules represent specific stages of processing, which makes it necessary for them to have specific responsibilities within the platform. The data transformation process should be extensible so that the processes can be suitable to work according to the characteristics of each type of data;
6) Keep the transmitted data's privacy;
7) Allow the extraction of knowledge from large volumes of data integrated to the platform;
8) Enable monetization, allowing the developers of the services to sell the data created in them.

Table I shows how the studies analyzed in Section II treat the requirements listed above for the proposed platform. Thus, it is clear to see that none of them defines an extensible transformation flow for processing the data in their architecture proposals, which is a requirement that is the main focus of this proposal. In this flow, we determine the steps required to process a group of data that is integrated to the platform in order to deliver them in the best way possible to be used in the development of new systems. Moreover, the extensibility of the defined steps allows these steps' processing are realized according to the characteristics of each type of data.

TABLE I. REQUIREMENTS ATTENDED BY THE RELATED WORKS.

Requirement	Works
Retrieve data from heterogeneous sources	[12], [8], [10], [9], [6], [11]
Create new services	[12], [8], [10], [9], [6], [11]
Support data aggregation	[8], [9], [6], [11]
Well-defined and extensible data processing	-
Allow the extraction of knowledge	-
Modular approach	[10], [9], [6], [11]
Pluggable approach	-
Maintains data privacy	[8]
Monetization	-

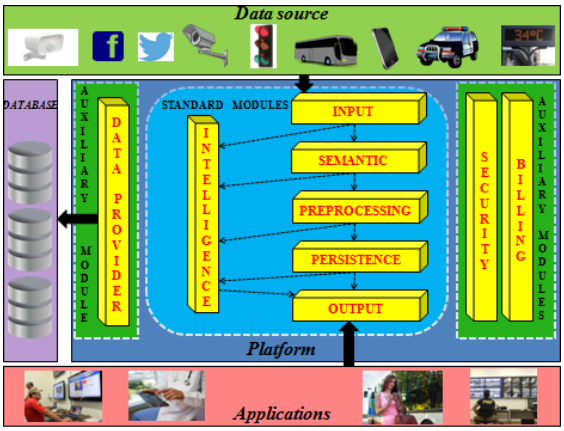


Figure 1. Proposed platform.

A. Architecture

Due to the requirements listed in Section III, we decided to carry out the implementation of the proposed architecture in the Open Services Gateway Initiative (OSGi) framework. This technology makes the development of modular Java softwares easier due to the fact that it provides many benefits related to manageability and maintainability [13], which is essential for this solution since the extensible transformation process thought for it aims to use the modular and "pluggable" approaches, allowing the extensions to be easily inserted and removed.

Figure 1 displays the platform proposed in this article. In Figure 1, we can see that it was planned in a way to support data from different sources, which are treated within it and then are made available, allowing the creation of new applications. In addition to this, it is possible to identify that when it comes to the database that should be used, it is flexible and supports the use of different types of database. The platform also has a set of **standard modules**, which are responsible for defining the steps of the extensible processing flow. Moreover, in the proposed solution, there is a set of **auxiliary modules** that increase the features that are important to it.

Each of these standard modules has its own specific responsibilities in the architecture and provide its basic behaviors. Furthermore, as shown in Figure 2, they are responsible for defining the extension points, allowing different implementations to be generated and "plugged" on to the platform.

The **specific modules** are responsible for implementing the processing tasks for each step of the extensible processing flow. Therefore, to add a source to the architecture, it is necessary to implement the specific modules that are able to handle the

 ISBN: 978-1-61208-438-1

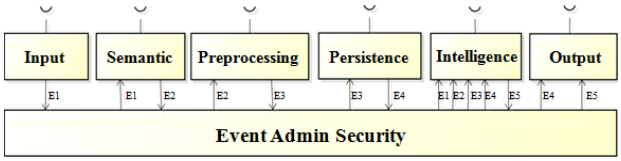


Figure 2. Extensible architecture.

type of data being inserted and then plug them to their relating standard modules.

In Figure 2, it is also possible to identify the existence of the **Event Admin Security** module, which is an extension of the Event Admin available in OSGi. This extension was carried out with the goal of adding an additional security requirement related to the access to messages transferred in this module. Thus, this capability ensures that only the architecture's standard modules can receive messages from the Event Admin, preventing the specific modules to interfere in their flow.

The standard modules work partly in a similar way and are only distinguished from each other in the processing step for which they are responsible. In general, a standard module receives a set of data. Then, it checks the specific modules that are interested in the type of data received and passes it to those who are allowed to access it. Thus, these specific modules process and return the data to the standard module which, finally, publishes it using the **Event Admin Security**. This way, the architecture has six standard modules, which are:

1) **Input**: the modules that are "plugged" on to the Input are responsible for integrating different data sources that exist in the cities to the proposed architecture;
2) **Semantic**: is responsible for receiving the data that is integrated in the Input and representing them in the format that the developer feels is most appropriate to the system that is being implemented;
3) **Preprocessing**: receives the data treated in Semantic and is responsible for filtering it;
4) **Persistence**: its tasks is to receive the preprocessed data and the primary responsibility of the modules that are "plugged" on to it is to store the received data in the architecture;
5) **Intelligence**: is responsible for receiving all of the data processed by the modules mentioned above. This way, the specific modules process this data and when their algorithms can identify any relevant knowledge, event 5 (E5) is published;
6) **Output**: is responsible for receiving the data processed by the Persistence and the Intelligence modules. Finally, each individual Output module provides access to the architecture's data, allowing new applications to be developed.

The proposed architecture also has three auxiliary modules: **Security**, which implements the policy of permissions to access data that are transferred within the architecture; **Billing**, which is responsible for accounting the messages that are accessed by the specific modules to enable later billing related to data consumption; and **Data Provider**, whose function is to carry out the management of the stored data and allow them to be accessed by the specific modules.

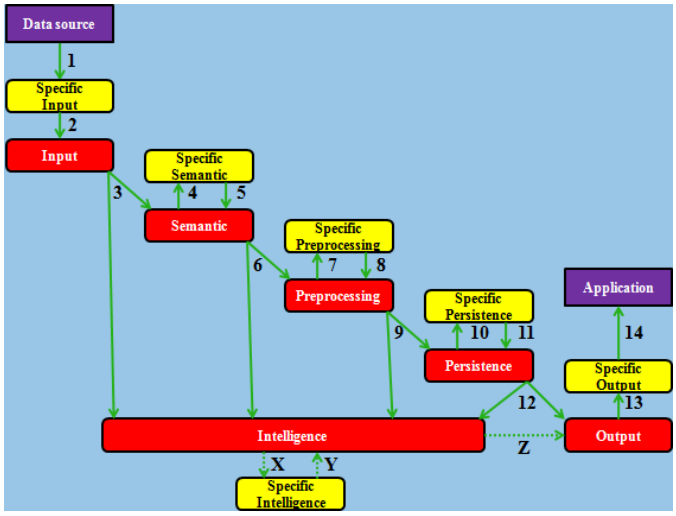


Figure 3. Steps of the data processing flow.

To perform the data aggregation process in a module, it is only necessary for it to have the set of permissions to access data from different services that lead to the compound service. Finally, the developer is not obliged to provide specific modules for all of the steps in the transformation flow. Thus, the platform will transfer the data to the next module of the flow when it is not possible to find, in a standard module, specific implementations responsible for working with the data type that was received.

B. Data flow in the platform

As defined in Figure 3, by making use of these six modules and using a simplified scenario where there is only one specific module "plugged" to each standard module, the basic extensible flow occurs through a set of 14 steps. First, the data is sent from the source to the **Specific Input** module responsible for receiving it (Step 1), which, in Step 2, forwards it to their relating standard module.

Then, in Step 3, the data is transferred from the Input to the next step (Semantic). Thus, this standard module forwards it to the specific module that is capable of working with it (Step 4). After it is received, the **Specific Semantic** module performs the first data transformation, since it is at that instant that it starts to be represented in the format chosen by the developer of the specific system. After that, in Step 5, ir is sent back to Semantic and then the data is published by this standard module (Step 6).

Preprocessing receives the data transferred in Step 6 and delivers it to the **Specific Preprocessing** module (Step 7), which performs a filtering process in which the set of data is subjected to a cleaning and selection process. After that, in Step 8, the filtered data returns to the Preprocessing module, which passes it forward (Step 9).

In Step 10, Persistence transfers the pre-processed data to the **Specific Persistence** module. After that, this specific module performs the data storage process and then, in Step 11, returns the last state of the problem's data to Persistence, which forwards it to the next step of the flow (Step 12).

Subsequently, Output passes the data to the specific module

 ISBN: 978-1-61208-438-1

(Step 13). Thus, in Step 14, the **Specific Output** module makes the data accessible to applications.

It is important to note that any of the specific modules can perform data aggregation in its processing tasks. In addition, Figure 3 also shows a knowledge discovery flow. In it, the Intelligence module receives all of the data delivered in steps 3, 6, 9, and 12 of the basic flow. Every time a set of data is received, the standard Intelligence module forwards it to the **Specific Intelligence** module (Step X) which processes it at all times in an attempt to identify any knowledge relevant to the problem. Therefore, when something meaningful is identified, the Specific Intelligence module returns the information to the standard Intelligence module which, in its turn, sends it to the Output module, which makes them available to the applications. Finally, for this flow, the letters X, Y and Z were used since it is not possible to predict the moment in which every one of the steps will be executed in the processing flow because they do not follow a sequential execution like the basic flow does.

IV. Case Study

This section describes a case study that aims to evaluate two behavioral aspects of the implemented platform: the easiness of the creation of specific modules (**Extensibility**) and the data processing capacity (**Performance**). Finally, the planning process and its description followed the guidelines set forth in [14][15][16].

A. Planning

This case study investigates the following research questions (RQ):

- **RQ1**: Is the platform extension process that is carried ou through the development of specific modules a simple activity?
- **RQ2**: Is the performance of the data flow's treatment process impaired in any way due to the existence of a set of steps for information processing?
- **RQ3**: Is the performance of the data flow's treatment process impaired when specific modules plugged to standard module are used?

The **subject** who used the platform that was proposed and implemented in this work was a developer with experience in the development of Java and OSGi applications. Moreover, the used **object** was an extension of the proposed platform depeloped to integrate data from a parking lot. Thus, in this scenario, we intended to access data from the server that stored the parking lot's information, process it using the platform's extensible flow and then make it available for the development of new applications.

The analysis units for this case study are: the implemented platform and its extension that enables to work with the parking lot's data. Thus, the platform's standard modules were evaluated regarding the performance of the data flow's transformation process. The extension used to handle the parking lot's resource, in its turn, was explored regarding the extensibility analysis and the evaluation of the performance of the data flow when specific modules are "plugged" on to standard modules.

B. Execution

To insert the data from the parking lot in the proposed platform, the implementation of specific modules to work with this source was generated. A priori, the additional module **Parking Model** was developed, which is used by all of the specific modules and whose responsibility is to mold in classes the data from the parking lot source.

Then, the **Parking Input** module was implemented, which integrates the data generated in the parking lot to the platform. Subsequently, the **Parking Semantic** was generated, which is responsible for representing such data in objects. Thereafter, the **Parking Preprocessing** module was developed, whose duties are to eliminate data duplication and select only the main data of the problem. In sequence, the **Parking Persistence** was implemented, which is only responsible for performing the received data storage step. In this study, the **Parking Intelligence** module was also developed, which only stores in a file the logs from all of events 1, 2, 3 and 4 sent in the **Event Admin Security**. This was important to confirm the sequence of the sent events. The **Parking Output** module is a Representational State Transfer (REST) module that works as a gateway for the parking lot's resource data processed in the architecture.

After the implementation of all of these modules, we moved on to the stage of evaluation of all of the **extensibility** and **performance** aspects of the platform. This evaluation was performed in a machine with Windows 8.1 operation system Single Langue 64-bit, Intel (R) Core (TM) i3-3227U CPU @ 1.90GHz and 3.87 GB of RAM processor.

Regarding the extensibility, we collected the amount of lines of code implemented in each of the specific modules of the parking lot's system, as shown in Table II. On this count, all of the lines of code in the source code's file were accounted for, including imports, statements, etc. In addition, we also counted the lines of code that are directly related to tasks that are necessary to "plug" these modules to the platform, as shown in Table III.

TABLE II. Lines of code of parking lot's system

Number of lines of code	
Parking Input	78
Parking Semantic	66
Parking Preprocessing	79
Parking Persistence	68
Parking Intelligence	78
Parking Output	61
Parking Model	56
Total	486

TABLE III. Number of lines of code in the parking lot system's modules (ignoring statements and general code)

Number of lines of code	
Parking Input	7
Parking Semantic	9
Parking Preprocessing	8
Parking Persistence	10
Parking Intelligence	9
Parking Output	12
Total	55

Regarding the performance, this requirement was evaluated

 ISBN: 978-1-61208-438-1

using the standard of measurement of the time it takes for the data to be transferred in the flow. This measure was calculated based on the time required for a message to be transferred from the input point to the end of the processing flow. For this purpose, the average time that it takes a certain amount of packages sent at once to go through all of the processing steps of the platform was calculated. In addition, for each amount of packages sent, ten samples were collected and their general average time was generated using (1).

$$a = \frac{\sum_{j=1}^{j=10} \frac{\sum_{i=1}^{i=p} ti}{p}}{10} \quad (1)$$

Where:

- ***a*** – represents the general average;
- ***p*** – represents the amount of packages received;
- ***ti*** – represents the transfer time for the ***i*** package.

C. Threats to validity

For the case study, four types of validity were evaluated:

- **Construct validity**: data capture for this case study's execution was performed using quantitative surveys related to factors analyzed for the implemented platform. Moreover, this process took place in a single machine, preventing changes in computer settings to compromise the values collected in the study;
- **Internal validity**: the features of the subject that performed the case study decreased the risk that factors related to inexperience in the development of Java and OSGi-based applications got out of control;
- **External validity**: programmers that are beginning to work with Java and OSGi can generate solutions with a larger amount of lines of code than those developed by the subject that performed the case study. Finally, the use of computer settings that are different from those specified in Section IV-B will influence the time it takes for messages to be processed by the platform;
- **Conclusion validity**: quantitative data that contributed to the platform evaluation process were used. Regarding performance data, they were collected in several samples in order to get an average, preventing that deviations that reflected only one specific time influence the outcome.

D. Answers to the research questions

This subsection answers the research questions raised in Section IV-A.

1) ***RQ1****:* The extension of the modules responsible for processing the data flow is a simple task, since it is only necessary to implement a small part of the code in order to plug them to the platform. As shown in Table II, in order to carry out the specific application of the six modules of the parking lot system, the implementation of 486 lines of code was necessary. However, by observing Table III, it is possible to note that less than 1/8 of the lines accounted for in Table II are directly responsible for providing the extension process defined in the standard modules.

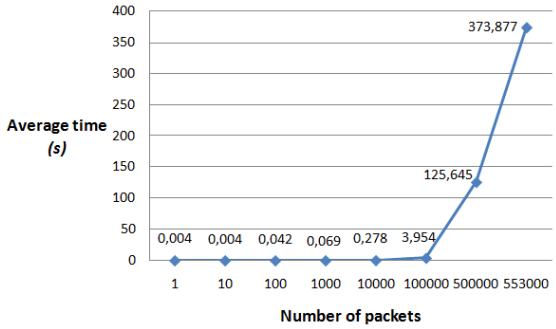


Figure 4. Average time to transport messages depending on the amount of packages (512 MB limit)

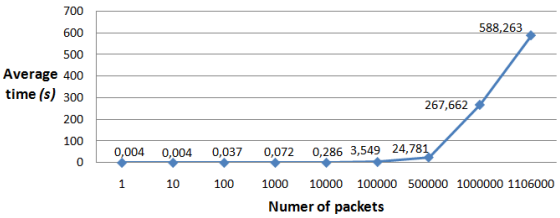


Figure 5. Average time to transport messages depending on the amount of packages (1,024 MB limit)

2) ***RQ2****:* By looking at the flow's transfer data, it is possible to realize that the data processing steps do not affect significantly the platform's performance. However, this feature depends on the settings of the computer in which it runs since, as shown in the results in Figure 4 and Figure 5, it is possible to note that the average transfer time for up to 100,000 packages received at the same time is stable, but when this number of messages is increased, the transfer time increases dramatically. Another factor that can prove this fact is the moment in which the memory limit was doubled. With this, the transfer time for the amount of 500,000 packages went down to 80.27% when compared to the 512 MB of RAM memory experiment. Furthermore, by providing the platform twice the RAM memory, the maximum number of packages supported also doubled, which shows that the amount of messages supported by Event Admin Security depends on the amount of memory available.

3) ***RQ3****:* By analyzing the graph shown in Figure 6, it is possible to note that the use of specific modules to perform the processing of the type of messages in the parking lot data causes a loss of performance in the delivery of packages when compared to the experiment shown previously in Figure 4. However, the average time only reaches very high values when the amount of messages received simultaneously is also very high, as can be seen in Figure 6, where up to the amount of 1,000 messages, the average transfer time is approximately 2 seconds.

V. Conclusion and Future Work

This article shows the details in definition, design and implementation of a platform that aims to integrate, transform and provide heterogeneous data generated in the urban environment. It has an extensible processing flow, which is important because it is not possible to predict how the different

 ISBN: 978-1-61208-438-1

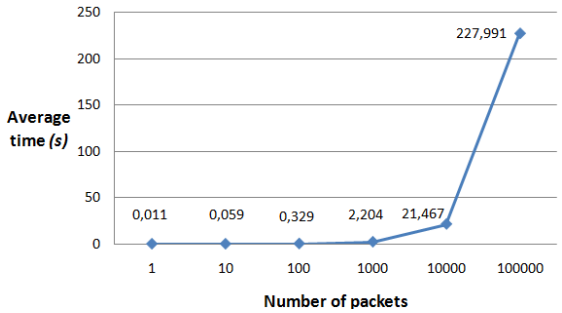


Figure 6. Average time to transport messages depending on the amount of packages using the parking lot's specific modules (512 MB limit)

types of data need to be processed in order to be delivered to the applications and because as time goes by, new types of data will emerge and will also need to be "plugged" to the platform. This way, the main contribution of this work and of the proposed platform was the creation of an extensible processing flow that allows data processing to be suitable to work according to the characteristics of each type of data existing in the ecosystem.

Through the extension that was performed in order to work with the parking lot scenario, it was possible to insert data from a first source into the platform. With the specific modules that were developed, it was possible to test the extensible processing flow, wherein they perform processing according to the characteristics of the parking lot's feature data. Furthermore, it was possible to attest the operation of the standard modules and the auxiliary modules defined in this work.

With the case study, it was possible to evaluate two important aspects related to the platform implementation proposal. The extensibility characteristic was a process that was easily carried out due to the fact that the standard modules made available the interfaces that define the behaviors associated with it, which makes the process of extension and "plugging" specific modules easy. Moreover, with the evaluation of the performance, there is a proof that the steps defined do not burden significantly the transfer of messages in the platform.

As future work, we intend to perform further case studies of the use of the platform, in which we aim to work with scenarios where there are different sources of data allowing their aggregation and also the development of multiple applications. Furthermore, there is the intent to evaluate other characteristics related to platform implementation, such as processing and distributed scalability. Moreover, we intended to make the cloud computing and big data concepts better in it. Finally, we aim to use the platform to manage a real environment where there are several devices, applications and the possibility of the emergence of new sources of data.

REFERENCES

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," Internet of Things Journal, vol. 1, no. 1, February 2014, pp. 22–32.

[2] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," in 12th Annual International Conference on Digital Government Research. ACM, June 2011, pp. 282–291.

[3] R. Giffinger, C. Fertner, H. Kramar, R. Kalasek, N. Pichler-Milanovic, and E. Meijers, "Smart cities: Ranking of european medium-sized cities," Centre of Regional Science (SRF), Tech. Rep., 2007.

[4] A. Mostashari, F. Arnold, M. Maurer, and J. Wade, "Citizens as sensors: The cognitive city paradigm," in 8th International Conference Expo on Emerging Technologies for a Smarter World (CEWIT). IEEE, November 2011, pp. 1–5.

[5] K. Su, J. Li, and H. Fu, "Smart city and the applications," in International Conference on Electronics, Communications and Control (ICECC). IEEE, September 2011, pp. 1028–1031.

[6] K. Gama, L. Touseau, and D. Donsez, "Combining heterogeneous service technologies for building an internet of things middleware," Computer Communications, vol. 35, no. 4, November 2012, pp. 405–417.

[7] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," ACM Computing Surveys, vol. 44, no. 3, June 2012, pp. 15:1–15:62.

[8] L. Anthopoulos and P. Fitsilis, "From digital to ubiquitous cities: Defining a common architecture for urban development," in Sixth International Conference on Intelligent Environments (IE). IEEE, July 2010, pp. 301–306.

[9] L. Filipponi, A. Vitaletti, G. Landi, V. Memeo, G. Laura, and P. Pucci, "Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors," in Fourth International Conference on Sensor Technologies and Applications. IEEE, July 2010, pp. 281–286.

[10] M. Blackstock, N. Kaviani, R. Leal, and A. Friday, "Magic broker 2: An open and extensible platform for the internet of things," in Internet of Things (IOT). IEEE, November 2010, pp. 1–8.

[11] B. Valente and F. Martins, "A middleware framework for the internet of things," in The Third International Conference on Advances in Future Internet. IARIA, 2011, pp. 139–144.

[12] F. Andreini, F. Crisciani, C. Cicconetti, and R. Mambrini, "A scalable architecture for geo-localized service access in smart cities," in Future Network and Mobile Summit (FutureNetw). IEEE, June 2011, pp. 1–8.

[13] P. Bakker and B. Ertman, Building Modular Cloud Apps with OSGi. USA: O Reilly, 2013.

[14] R. K. Yin, Case Study Research: Design and Methods. SAGE Publications, 2003.

[15] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," IEEE software, vol. 12, no. 4, July 1995, pp. 52–62.

[16] P. Runeson and M. Host, "Guidelines for conducting and reporting case study research in software engineering," Empirical software engineering, vol. 14, no. 2, April 2009, pp. 131–164.

 ISBN: 978-1-61208-438-1

Improving the Application of Agile Model-based Development: Experiences from Case Studies

K. Lano
H. Alfraihi
S. Yassipour-Tehrani
Dept. of Informatics
King's College London
London, UK
Email: kevin.lano@kcl.ac.uk, hessa.alfraihi@kcl.ac.uk,
s.yassipour-tehrani@kcl.ac.uk

H. Haughton
Holistic Risk Solutions Ltd
Croydon, UK

Email: howard_haughton@btinternet.com

***Abstract*—Agile model-based development has the potential to combine the benefits of both agile and model-based development (MBD) approaches: rapid automated software generation, lightweight development processes and direct customer involvement. In this paper, we analyse three application case studies of agile MBD, and we identify the factors which have contributed to the success or failure of these applications. We propose an improved agile MBD approach, and give guidelines on its application, in order to increase the effectiveness and success rate of applications of agile MBD.**

Keywords — Model-based development (MBD); Model-driven development (MDD); Agile development.

I. INTRODUCTION

Agile development and model-based development (MBD) are two alternative software development approaches which have been devised to address the 'software crisis' of software project failures and excessive development costs. Both approaches have been adopted by industry to a certain extent, and with some evidence of success. But both approaches also have drawbacks and limitations, which have restricted their uptake.

The idea of combining the approaches into an 'agile MBD' approach has been explored, with the intention that such an approach would avoid the deficiencies of the individual methods [5][12]. In some ways, agile and MBD development approaches are compatible and complementary. For example:

- Both agile development and MBD aim to reduce the gap between requirements analysis and implementation, and hence the errors that arise from incorrect interpretation or formulation of requirements. Agile development reduces the gap by using short incremental cycles of development, and by direct involvement of the customer during development, whilst MBD reduces the gap by automating development steps.
- Executable models (or models from which code can be automatically generated) of MBD potentially serve as a good communication medium between developers and stakeholders, supporting the collaboration which is a key element of agile development.
- Automated code generation accelerates development, in principle, by avoiding the need for much detailed manual low-level coding.
- The need to produce separate documentation is reduced or eliminated, since the executable model is its own documentation.

On the other hand, the culture of agile development is heavily code-centric, and time pressures may result in fixes and corrections being applied directly to generated code, rather than via a reworking of the models, so that models and code become divergent. A possible corrective to this tendency is to view the reworking of the model to align it to the code as a necessary 'refactoring' activity to be performed as soon as time permits. We have followed this approach in several time-critical MBD applications.

Tables I and II summarise the parallels and conflicts between MBD and Agile development.

TABLE I. ADAPTIONS OF AGILE DEVELOPMENT PRACTICES FOR MBD

Practice	*Adaption*
Refactoring for quality improvement	Use model refactoring, not code refactoring
Test-based validation	(i) Generate tests from models (ii) Correct-by-construction code generation
Rapid iterations of development	Rapid iterations of modeling + Automated code generation
No documentation separate from code	Models are both code and documentation

TABLE II. CONFLICTS BETWEEN AGILE DEVELOPMENT AND MBD

Conflict	*Resolutions*
Agile is oriented to source code, not models	(i) Models as code (ii) Round-trip engineering (iii) Manual re-alignment
Agile focus on writing software, not documentation	Models are both documentation and software
Agile's focus on users involvement in development, versus MBD focus on automation	Active involvement of users in conceptual and system modelling

There are therefore different ways in which agile development and MBD can be combined, and the current agile MBD methods adopt different approaches for this integration. In this paper, we examine one possible approach for combining

 ISBN: 978-1-61208-438-1

agile and MBD, based on the Unified Modeling Language Rigorous Specification, Design and Synthesis (UML-RSDS) formalism and tools [9], which we summarise in Section II. This is compared with other agile MBD approaches in Section III. We then report results from three case studies using the UML-RSDS approach (Sections IV,V,VI), and in Section VII, we summarise the lessons learnt from these applications and give guidelines for improving the approach. Section VIII gives conclusions.

II. UML-RSDS

UML-RSDS is based on the class diagram, use case and Object Constraint Language (OCL) notations of UML. System specifications can be written in these notations, and then a design expressed using UML activities can be automatically synthesised from the specifications. Finally, executable code in several alternative languages (Java, C# and C++) can be automatically synthesised from the design [8]. Both structural and behavioural code is synthesised, and a complete executable is produced. The aim of the approach is to automate code production as much as possible, including code optimisation, so that system specifications can be used as the focus of development activities. Some configuration of the design choices can be carried out manually. The system construction process supported by UML-RSDS is shown in Figure 1.

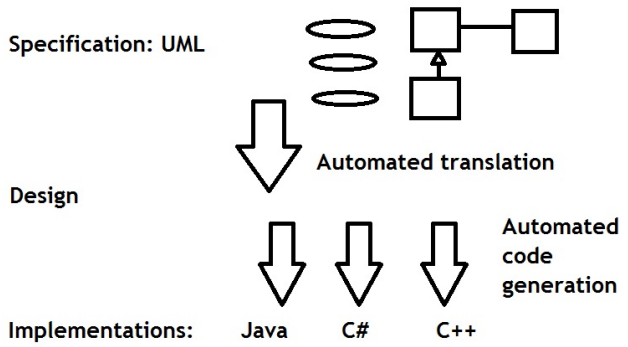


Figure 1. UML-RSDS software production process

An example specification of behaviour in UML-RSDS, from case study 2, is the following use case postcondition, which checks the GP data for duplicated patients (task 1b):

```
p : PatientGP & Id < p.Id &
name1 = p.name1 & name2 = p.name2 &
dob = p.dob & isMale = p.isMale   =>
    ("Patients " + self + " and " + p +
     " seem to be duplicates")->display()
```

This iterates over *self* : *PatientGP*, and displays a warning message for each other patient *p* that has the same name, date of birth and gender as *self*, but a different id value.

The UML-RSDS approach supports agile development, with the options (ii) (correct-by-construction code generation) and (i) (models as code) from Tables I and II being used to combine MBD and agile concepts.

III. Related work

A small number of other agile MBD approaches have been formulated and applied: Executable UML (xUML) [13]; Sage [7]; MDD System Level Agile Process (MDD-SLAP) [18]; Hybrid MDD [4]. Both xUML and UML-RSDS use the principle that "The model is the code", and support incremental system changes via changes to the specification. There is a clearly-defined process for incremental revision in UML-RSDS, MDD-SLAP and Hybrid MDD. MDD-SLAP and Hybrid MDD define explicit integration processes for combining synthesised and hand-crafted code.

Explicit verification processes are omitted from Sage and Hybrid MDD. In MDD-SLAP, simulation and test-driven modelling is used for validation and verification [18]. Some support for formal validation and verification is provided by xUML and UML-RSDS: the iUML tool for xUML has support for simulation, and UML-RSDS provides correctness analysis via a translation to the B formal method. By automating code generation, agile MBD approaches should improve the reliability and correctness of code compared to manual-coding development. All the approaches are focussed on one-way forward engineering, and do not support round-trip engineering, which means that synchronisation of divergent code and models is a manual process.

UML-RSDS and xUML are based on modelling using the standard UML model notations, with some variations (action language in the case of xUML, use cases specified by constraints in UML-RSDS), and on following a general MDA process: Computation-independent Model (CIM) to Platform-independent Model (PIM) to Platform-specific Model (PSM) to code. Platform modelling is explicitly carried out in xUML but not in UML-RSDS, which restricts developers to Java-like languages for the executable code. Sage uses variants of UML models oriented to reactive system definition using classes and agents. These include environmental, design, behavioural and runtime models. An executable system is produced by integration of these models. MDD-SLAP maps MDD process activities (requirements analysis and high-level design; detailed design and code generation; integration and testing) into three successive sprints used to produce a new model-based increment of a system. Hybrid MDD envisages three separate teams operating in parallel: an agile development team hand-crafting parts of each release; a business analyst team providing system requirements and working with a MDD team to produce domain models. The MDD team also develops synthesised code. MDD-SLAP and Hybrid MDD have the most elaborated development processes. The survey of [5] identifies that Scrum-based approaches such as MDD-SLAP are the most common in practical use of agile MBD (5 of the seven cases examined), with XP also often used (4 of 7 cases). The agile MDD approach in the case of [15] used Scrum and Kanban.

IV. Case study 1: FIXML code generation

This case study was based on the problem described in [15]. Financial transactions can be electronically expressed using formats, such as the Financial Information eXchange (FIX) format. New variants/extensions of such message formats can be introduced, which leads to problems in the maintenance of end-user software: the user software, written in various programming languages, which generates and processes financial transaction messages will need to be updated to the latest version of the format each time it changes. In [15], the author proposed to address this problem by automatically synthesising

 ISBN: 978-1-61208-438-1

program code representing the transaction messages from a single XML definition of the message format, so that users would always have the latest code definitions available. For this case study we restricted attention to generating Java, C# and C++ class declarations from messages in FIXML 4.4 format [2][3].

The solution transformation should take as input a text file of a message in XML FIXML 4.4 Schema format, and produce as output corresponding Java, C# and C++ text files representing this data.

The problem is divided into the following use cases:

1) Map data represented in an XML text file to an instance model of the XML metamodel.
2) Map a model of the XML metamodel to a model of a suitable metamodel for the programming language/languages under consideration. This has subtasks: 2a. Map XML nodes to classes; 2b. Map XML attributes to attributes; 2c. Map subnodes to object instances.
3) Generate program text from the program model.

In principle, these use cases could be developed independently, although the subteams or developers responsible for use cases 2 and 3 need to agree on the programming language metamodel(s) to be used.

The problem was set as the assessed coursework (counting for 15% of the course marks) for the second year undergraduate course "Object-oriented Specification and Design" (OSD) at King's College in 2013. It was scheduled in the last four weeks at the end of the course. OSD covers UML and MBD and agile development at an introductory level. Students also have experience of team working on the concurrent Software Engineering Group project (SEG). Approximately 120 students were on the course, and these were divided into 12 teams of 10 students each, using random allocation of students to teams.

The students were instructed to use UML-RSDS to develop the three use cases of the problem, by writing specifications using the UML-RSDS tools and generating code from these specifications. They were also required to write a team report to describe the process they followed, their team organisation, and the system specification. The case study involves research into FIXML, XML, UML-RSDS and C# and C++, and carrying out the definition of use cases in UML-RSDS using OCL. None of these topics had been taught to the students. Scrum, XP, and an outline agile development approach using UML-RSDS had been taught, and the teams were recommended to appoint a team leader. A short (5 page) requirements document was provided, and links to the UML-RSDS tools and manual. The XML metamodel was provided, and a UML-RSDS library to parse XML was also given to the students to use. Each week there was a one hour timetabled lab session where teams could meet and ask for help from postgraduate students who had some UML-RSDS knowledge. The outcome of the case study is summarised in Table III.

Examples of good practices included:

- Division of a team into sub-teams with sub-team leaders, and separation of team roles into researchers and developers (teams 8, 11).
- Test-driven development (teams 8, 9).

TABLE III. CASE 1 RESULTS

Teams	*Mark range*	*Result*
5, 8, 9, 10	80+	Comprehensive solution and testing, well-organised team
12	80+	Good solution, but used manual coding, not UML-RSDS
4, 7, 11	70-80	Some errors/incompleteness
2, 3, 6	50-60	Failed to complete some tasks
1	Below 40	Failed all tasks, group split into two.

- Metamodel refactoring, to integrate different versions of program metamodels for Java, C# and C++ into a single program metamodel.

Exploratory and evolutionary prototyping were used by most teams as their main development process. However, most teams experienced substantial obstacles in the project, due to (i) problems with the interface of the UML-RSDS tools, which did not conform to the usual style of development environment (such as NetBeans) which the students were familiar with; (ii) problems understanding and using the MBD executable model concept. Only four teams managed to master the development approach, others either reverted to manual coding or produced incomplete solutions. The total effort expended by successful MBD teams was not in excess of that expended by the successful manual coding team, which suggests that the approach can be feasible even in adverse circumstances.

V. CASE STUDY 2: ELECTRONIC HEALTH RECORDS (EHR) ANALYSIS AND MIGRATION

This case study was the OSD assessed coursework for 2014. It was intended to be somewhat easier than the 2013 coursework. Approximately 140 second year undergraduate students participated, divided into 14 teams of 9 or 10 members. Students were allocated randomly to teams.

There were three required use cases: (1) to analyse a dataset of GP patient data conforming to the class diagram of Figure 2 for cases of missing names, address, or other feature values; (2) to display information on referrals and consultations in date-sorted order; (3) to integrate the GP patient data with hospital patient data conforming to the class diagram of Figure 3 to produce an integrated dataset conforming to a third class diagram (gpmm3).

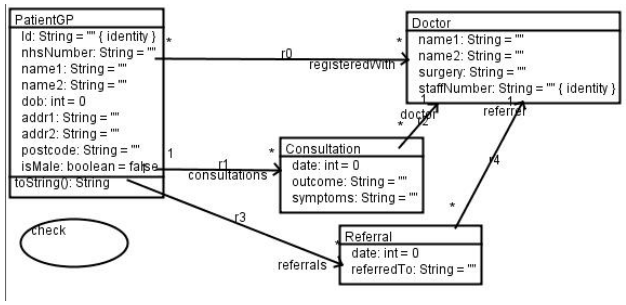


Figure 2. GP patient model gpmm1

Table IV summarises the use cases and their subtasks.

As with case study 1, the teams were required to use UML-RSDS to develop the system, and to record their organisation and results in a report. Teams were advised to select a leader,

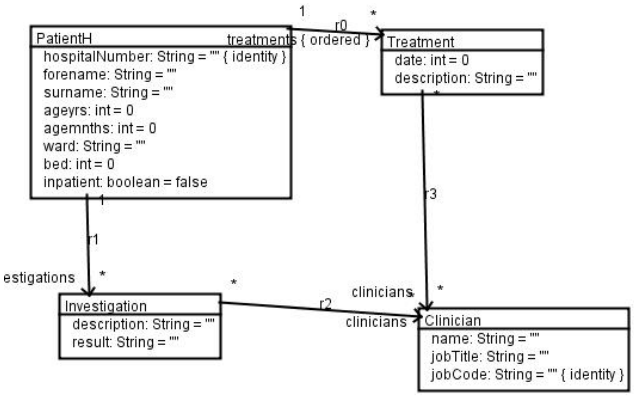


Figure 3. Hospital patient model gpmm2

TABLE IV. USE CASES FOR EHR ANALYSIS/MIGRATION

Use case	*Subtasks*	*Models*
1. Analyse data	1a. Detect missing data in GP dataset	gpmm1
	1b. Detect duplicate patient records	gpmm1
2. View data	2a. Display consultations of each GP patient, in date order	gpmm1
	2b. Display referrals of each GP patient, in date order	gpmm1
3. Integrate data	Combine gpmm1, gpmm2 data into gpmm3	gpmm1, gpmm2, gpmm3

TABLE V. OUTCOMES OF EHR CASE STUDY

Team	*Training time*	*Technical outcome*	*Agile process*	*Activities, issues, process*
1	$>$ 1 week	8/10	8/10	Disorganised and individual working
2	1 week	9/10	8/10	No experience of large teams
3	$>$ 1 week	8/10	9/10	Used pair modelling, proactive time planning
4	1 week	7/10	8/10	No leader. Parallel working
5	1 week	9/10	8/10	Lead developers
6	1 week	8/10	9/10	Used Scrum, subteam modelling, model refactoring
7	1 week	8/10	9/10	Risk analysis, paired modelling
8	1 week	9/10	9/10	Small team modelling. Lead developers trained team
9	$>$ 1 week	7/10	7/10	No leader, disorganised
10	1 week	8/10	8/10	Detailed planning, scheduling. Lead developers trained team
11	1 week	9/10	9/10	Used XP
12	$>$ 1 week	7/10	5/10	Team split into 2
13	2 weeks	8/10	8/10	Strong leadership
14	2 weeks	0/10	0/10	Failed to work as a team

and to apply an agile development process, although a specific process was not mandated. A short (2 page) requirements document was provided, and links to the UML-RSDS tools and manual. The three EHR models were provided. Each week there was a one hour timetabled lab session where teams could meet and ask for help from postgraduate students who had some UML-RSDS knowledge.

A. Outcomes

Of the 14 teams, 13 successfully applied the tools and an agile methodology to produce a working solution. Table V shows the characteristics of the different team solutions. Training time refers to the time needed to learn MBD using UML-RSDS.

Typically the teams divided into subteams, with each subteam given a particular task to develop, so that a degree of parallel development could occur, taking advantage of the independence of the three use cases. Most groups had a defined leader role (this had been advised in the coursework description), and the lack of a leader generally resulted in a poor outcome (as in teams 1, 4, 9, 12, 14). As with case study one, exploratory and evolutionary prototyping of specifications was used by the teams.

The key difficulties encountered by most teams were:

- Lack of prior experience in using UML.
- The unfamiliar style of UML-RSDS compared to tools such as Visual Studio, Net Beans and other development environments.
- Conceptual difficulty with the idea of MBD and the use of OCL to specify system functionality.
- Inadequate user documentation for the tools – in particular students struggled to understand how the tools were supposed to be used, and the connection between the specifications written in the tool and the code produced.
- Team management and communication problems due to the size of the teams and variation in skill levels and commitment within a team.

Nonetheless, in 12 of 14 cases the student teams overcame these problems. Two teams (12 and 14) had severe management problems, resulting in failure in the case of team 14.

The teams were almost unanimous in identifying that they should have committed more time at the start of the project to understand the tools and the MBD approach. This is a case where the agile principle of starting development as soon as possible needs to be tempered by the need for adequate understanding of a new tool and development technique.

Factors which seemed particularly important in overcoming problems with UML-RSDS and MBD were:

- The use of 'lead developers': a few team members who take the lead in mastering the tool/MBD concepts and who then train their colleagues. This spreads knowledge faster and more effectively than all team individuals trying to learn the material independently. Teams that used this approach had a low training time of 1 week, and achieved an average technical score of 8.66, versus 7.18 for other teams. This difference is statistically significant at the 4% level (removing team 14 from the data).
- Pair-based or small team modelling, with subteams

 ISBN: 978-1-61208-438-1

of 2 to 4 people working around one machine. This seems to help to identify errors in modelling which individual developers may make, and additionally, if there is a lead developer in each sub-team, to propagate tool and MBD expertise. Teams using this approach achieved an average technical score of 8.25, compared to 7.2 for other teams. This difference is however not statistically significant if team 14 is excluded.

Teams using both approaches achieved an average technical score of 9, compared to those using just one (8.2) or none (6.9).

Another good practice was the use of model refactoring to improve an initial solution with too complex or too finely-divided use cases into a solution with more appropriate use cases.

The impact of poor team management and the lack of a defined process seems more significant for the outcome of a team, compared to technical problems. The Pearson correlation coefficient of the management/process mark of the project teams with their overall mark is 0.91, suggesting a strong positive relation between team management quality and overall project quality. Groups with a well-defined process and team organisation were able to overcome technical problems more effectively than those with poor management. Groups 3, 5, 7, 11 and 13 are the instances of the first category, and these groups achieved an average of 8.4/10 in the technical score, whilst groups 1, 4, 9, 12 and 14 are the instances of the second category, and these groups achieved an average of 5.8/10 in the technical score. An agile process seems to be helpful in achieving a good technical outcome: the correlation of the agile process and technical outcome scores in Table V is 0.93.

The outcomes of this case study were better than for the first case study: the average mark was 79% in case study 2, compared to 67.5% for case study 1. This appears to be due to three main factors: (i) a simpler case study involving reduced domain research and technical requirements compared to case study 1. In particular there was no need to understand and use an external library such as the XML parser; (ii) improvements to the UML-RSDS tools; (iii) stronger advice to follow an agile development approach.

In conclusion, this case study illustrated the problems which may occur when industrial development teams are introduced to MBD and MBD tools for the first time. The positive conclusions which can be drawn are that UML-RSDS appears to be an approach which quite inexperienced developers can use successfully for a range of tasks, even with limited access to tool experts, and that the difficulties involved in learning the tools and development approach are not significantly greater than those that could be encountered with any new SE environment or tools.

VI. Case study 3: Collateralized Debt Obligations risk estimation

This case study concerns the risk evaluation of multiple-share financial investments known as *Collateralized Debt Obligations* (CDO), where a portfolio of investments is partitioned into a collection of sectors, and there is the possibility of contagion of defaults between different companies in the same sector [1][6]. Risk analysis of a CDO contract involves computing the overall probability $P(S = s)$ of a financial loss s based upon the probability of individual company defaults and the probability of default infection within sectors.

Both a precise (but very computationally expensive) and an approximate version of the loss estimation function $P(S = s)$ were required. The case study was carried out in conjunction with a financial risk analyst, who was also the customer of the development. Implementations in Java, C# and C++ were required.

The required use cases and subtasks are given in Table VI. Use case 3 depends upon tasks 2a and 2b of use case 2. Unlike

TABLE VI. USE CASES FOR CDO RISK ANALYSIS

Use case	*Subtasks*	*Description*
1. Load data		Read data from a .csv spreadsheet
2. Calculate Poisson approximation of loss function	2a Calculate probability of no contagion	
	2b. Calculate probability of contagion	
	2c. Combine 2a, 2b	
3. Calculate precise loss function		
4. Write data		Write data to a .csv spreadsheet

case studies 1 and 2, team management was not a problem because this was a single-developer project. In addition the developer was an expert in UML-RSDS. Therefore the focus of interest in this case study is how effectively agile development with UML-RSDS can be used for this domain.

First, a phase of research was needed to understand the problem and to clarify the actual computations required. Then tasks 2a, 2b and 2c were carried out in a first development iteration, as these were considered more critical than use cases 1 or 4. Exploratory and evolutionary prototyping were used. Then, the use case 3 was performed in development iteration 2, and finally use cases 1 and 4 – which both involved use of manual coding – were scheduled to be completed in a third development iteration. A further external requirement was introduced prior to this iteration: to handle the case of cross-sector contagion. This requirement was then scheduled in the third iteration, and tasks 1 and 4 in a fourth iteration.

Figure 4 shows the class diagram of the solution produced at the end of the first development iteration.

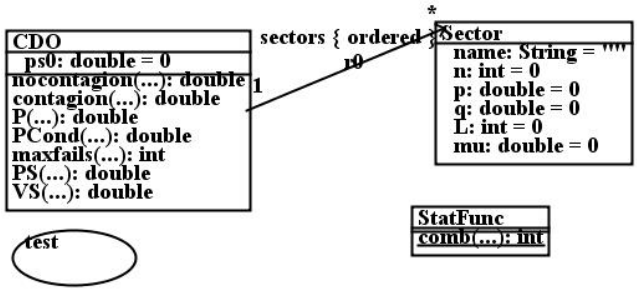


Figure 4. CDO version 1 system specification

The following agile development techniques were employed:

- Refactoring: the solutions of 2a and 2b were initially expressed as operations *nocontagion*, *contagion* of the CDO class (Figure 4). It was then realised that they would be simpler and more efficient if defined as Sector operations. The refactoring Move Operation was used. This refactoring did not affect the external interface of the system.
- Customer collaboration in development: the risk analyst gave detailed feedback on the generated code as it was produced, and carried out their own tests using data such as the realistic dataset of [6].

It was originally intended to use external hand-coded and optimised implementations of critical functions such as the combinatorial function $comb(int\ n, int\ m)$. However, this would have resulted in the need for multiple versions of these functions to be coded, one for each target implementation language, and would also increase the time needed for system integration. It was found instead that platform-independent specifications could be given in UML-RSDS which were of acceptable efficiency.

The initial efficiency of the approximate solution was too low, with calculation of $P(S = s)$ for all values of $s \leq 20$ on the test data of [6] taking over 2 minutes on a standard Windows 7 laptop. To address this problem, the recursive operations and other operations with high usage were given the stereotype $\ll$ *cached* $\gg$ to avoid unnecessary recomputation. This stereotype means that operations are implemented using the *memoisation* technique of [14] to store previously-computed results. Figure 5 shows the refactored system specification at the end of the third development iteration.

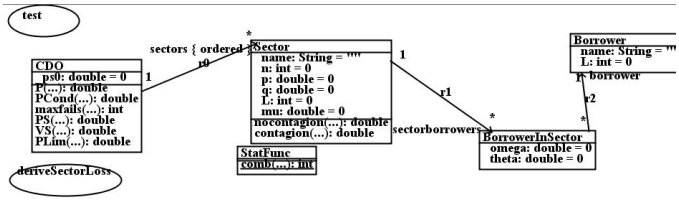


Figure 5. CDO version 3 system specification

Table VII shows the improvements in efficiency which memoisation provides, and the results for generated code in other language versions. The approximate version of $P(S = s)$ is compared.

TABLE VII. EXECUTION TIMES FOR CDO VERSIONS

Version	*Execution time for first 20 $P(S = s)$ calls*	*Execution time for first 50 $P(S = s)$ calls*
Unoptimised Java	121s	–
Optimised Java	32ms	93ms
C#	10ms	20ms
C++	62ms	100ms

Our experiences on this case study illustrate the UML-RSDS principles:

- Optimisation and refactoring should be carried out at the specification level in a platform-independent manner where possible, not at the code level.
- The scope of MBD should be extended as far as possible across the system development, reducing the scope of manual coding and integration wherever possible.

In conclusion, this case study showed that a successful outcome is possible for agile MBD in the highly demanding domain of computationally-intensive financial applications. A generic MBD tool, UML-RSDS, was able to produce code of comparable efficiency to existing hand-coded and highly optimised solutions.

VII. Guidelines for an improved agile MBD process

The case studies have identified the need for a well-defined agile MBD process for using UML-RSDS, and some techniques for improving the adoption and application of UML-RSDS, in addition to necessary technical improvements in the tools. In general it was found that a development approach using exploratory prototyping (of the system specification) at the initial stages, and evolutionary prototyping at later stages, was effective.

The following guidelines for adoption and application of agile MBD are proposed, on the basis of our experiences in the presented and other case studies:

- **Utilise lead developers** When introducing MBD to a team inexperienced in its use, employ a small number of team members – especially those who are most positive about the approach and who have appropriate technical backgrounds – to take the lead in acquiring technical understanding and skill in the MBD approach. The lead developers can then train their colleagues.
- **Use paired or small team modelling** Small teams working together on a task or use case can be very effective, particularly if each team contains a lead developer, who can act as the technical expert. It is suggested in [18] that such teams should also contain a customer representative.
- **Use a clearly defined process and management structure** The development should be based on a well-defined process, such as XP, Scrum, or the MBD adaptions of these given in this paper or by MDD-SLAP and Hybrid MDD. A team leader who operates as a facilitator and co-ordinator is an important factor, the leader should not try to dictate work at a fine-grained level, but instead enable sub-teams to be effective, self-organised and to work together.
- **Refactor at specification level** Refactor models, not code, to improve system quality and efficiency.
- **Extend the scope of MBD** Encompassing more of the system into the automated MBD process reduces development costs and time.

The first three of these are also recommended as good practices for agile development in general [10][11][17].

A detailed agile MBD process for UML-RSDS can be based upon the MDD-SLAP process. Each development iteration is split into three phases (Figure 6):

- **Requirements and specification:** Identify and refine the iteration requirements from the iteration backlog,

and express new/modified functionalities as system use case definitions. Requirements engineering techniques such as exploratory prototyping and scenario analysis can be used. This stage corresponds to the Application requirements sprint in MDD-SLAP. Its outcome is an iteration backlog with clear and detailed requirements for each work item.
If the use of MBD is novel for the majority of developers in the project team, assign lead developers to take the lead in acquiring technical skills in MBD and UML-RSDS.

- **Development, verification, code generation:** Subteams allocate developers to work items and write unit tests for their assigned use cases. Subteams work on their items in development iterations, using techniques such as evolutionary prototyping, in collaboration with stakeholder representatives, to construct detailed use case specifications. Formal verification at the specification level can be used to check critical properties. Reuse opportunities should be regularly considered, along with specification refactoring.
Daily Scrum-style meetings can be held within subteams to monitor progress, update plans and address problems. Techniques such as a Scrum board and burndown chart can be used to manage work allocation and progress. The phase terminates with the generation of a complete code version incorporating all the required functionalities from the iteration backlog.
- **Integration and testing:** Do regular full builds, testing and integration in an integration iteration, including integration with other software and manually-coded parts of the system.

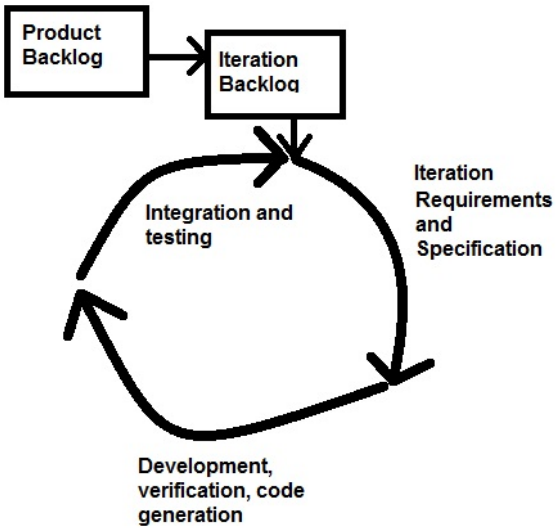


Figure 6. UML-RSDS process

VIII. CONCLUSIONS

We have analysed the process and outcomes of three case studies of MBD and agile development, involving a total of over 250 developers. From these cases, we have identified guidelines for the use of agile MBD, and an improved agile MBD process for UML-RSDS. In future work, we will develop a systematic evaluation framework for agile MDD application, and investigate extensions of our agile MDD process.

REFERENCES

[1] M. Davis and V. Lo, "Infectious Defaults", Quantitative Finance, vol. 1, no. 4, 2001, pp. 382–387.

[2] http://fixwiki.org/fixwiki/FPL:FIXML_Syntax. Accessed 11.9.2015.

[3] http://www.fixtradingcommunity.org. Accessed 11.9.2015.

[4] G. Guta, W. Schreiner, and D. Draheim, "A lightweight MDSD process applied in small projects", Proceedings 35th Euromicro conference on Software Engineering and Advanced Applications, IEEE, 2009, pp. 255-258.

[5] S. Hansson, Y. Zhao, and H. Burden, "How MAD are we?: Empirical evidence for model-driven agile development", XM Workshop, MODELS 2014, 2014, pp. 2-11.

[6] O. Hammarlid, "Aggregating sectors in the infectious defaults model", Quantitative Finance, vol. 4, no. 1, 2004, pp. 64–69.

[7] J. Kirby, "Model-driven Agile Development of Reactive Multi-agent Systems", COMPSAC '06, 2006, pp. 297–302.

[8] K. Lano and S. Kolahdouz-Rahimi, "Constraint-based specification of model transformations", Journal of Systems and Software, vol. 88, no. 2, February 2013, pp. 412–436.

[9] K. Lano, The UML-RSDS manual, http://www.dcs.kcl.ac.uk/staff/kcl/umlrsds.pdf, 2015.

[10] L. Lavazza, S. Morasca, D. Taibi, and D. Tosi, "Applying Scrum in an OSS Development Process: an Empirical Evaluation", in 11th International Conference XP 2010, 2010, pp. 147–159.

[11] R. C. Martin, "Agile Software Development: Principles, Patterns and Practices", Prentice Hall, 2003.

[12] R. Matinnejad, "Agile Model Driven Development: an intelligent compromise", 9th International Conference on Software Engineering Research, Management and Applications, 2011, pp. 197–202.

[13] S. Mellor and M. Balcer, "Executable UML: A foundation for model-driven architectures", Addison-Wesley, Boston, 2002.

[14] D. Michie, "Memo functions and machine learning", Nature, vol. 218, 1968, pp. 19–22.

[15] M. B. Nakicenovic, "An Agile Driven Architecture Modernization to a Model-Driven Development Solution", International Journal on Advances in Software, vol 5, nos. 3, 4, 2012, pp. 308–322.

[16] K. Schwaber and M. Beedble, "Agile software development with Scrum", Pearson, 2012.

[17] D. Taibi, P. Diebold, and C. Lampasona, "Moonlighting Scrum: an agile method for distributed teams with part-time developers working during non-overlapping hours", in ICSEA 2013, pp. 318–323.

[18] Y. Zhang and S. Patel, "Agile model-driven development in practice", IEEE Software, vol. 28, no. 2, 2011, pp. 84–91.

 ISBN: 978-1-61208-438-1

Metrics Framework for Cycle-Time Reduction in Software Value Creation

Adapting Lean Startup for Established SaaS Feature Developers

Pasi Tyrväinen, Matti Saarikallio
Agora Center, Department of CS and IS
University of Jyväskylä, Finland
pasi.tyrvainen@jyu.fi, matti.saarikallio@gmail.com

Timo Aho, Timo Lehtonen, Rauno Paukeri
Solita plc
Tampere, Finland
{timo.aho, timo.lehtonen, rauno.paukeri}@solita.fi

Abstract— **Agile software development methodologies driving cycle-time reduction have been shown to improve efficiency, enable shorter lead times and place a stronger focus on customer needs. They are also moving the process development focus from cost-reduction towards value creation. Optimizing software development based on lean and agile principles requires tools and metrics to optimize against. We need a new set of metrics that measure the process up to the point of customer use and feedback. With these we can drive cycle time reduction and improve value focus. Recently the lean startup methodology has been promoting a similar approach within the startup context. In this paper, we develop and validate a cycle-time-based metric framework in the context of the software feature development process and provide the basis for fast feedback from customers. We report results on applying three metrics from the framework to improve the cycle-time of the development of features for a SaaS service.**

Keywords-metrics framework; cycle-time; agile; software engineering process; lean startup; feedback; SaaS.

I. INTRODUCTION

The software engineering (SWE) process has traditionally been managed on a cost basis by measuring programmer effort spent per lines of code, function point or requirement. These metrics have also been used to guide software process improvement. In order to align more with business strategy and value production the focus has shifted more towards value creation instead of cost reduction. For example, value-based SWE [1], software value-map [2] and a special issue on return on investment (ROI) in IEEE Software [3] have explored value in software development. As a reaction to move away from a cost-reduction focus, the recent goal of lean thinking has been to optimize for perceived customer value [4]. Thus, we can say that leadership approach for the software development process is moving from a cost focus to a value focus.

Measuring the value of application software and cloud services is difficult to do before it is in use, as you need to consider the value of the software for the potential users, the business value for the firm developing it and the value for other stakeholders [1][5][6]. The current theories of value do not present a simple way of assessing customer value [7]. Although companies put a great amount of effort into increasing customers' perceived value in the product development process, determining how and when value is added is still a challenge even in marketing and management sciences. [7] Further, the software engineering metrics are measuring attributes of the software development process (e.g., cost, effort, quality) while these metrics remain disconnected from the attributes and metrics developed for measuring value (see Table I). Various approaches have been developed to overcome this gap [1][5][6][8][9][10][11][12][13][14][15][16] without any major break-through.

The software engineering community has adopted an iterative approach to software development in form of Scrum [17], XP [18] and other agile [19] methods. These promote fast cycle user interaction and development process to keep the effort focused on customer needs based on fast customer feedback either interactively or through analysis of service use behavior. The startup community has adopted a similar approach and commonly uses the lean startup cycle [20] to evaluate the hypothesis of customer needs using the build-measure-learn cycle, which is repeated to improve customer acceptability of the offering and the business value of the startup. The common theme of these approaches is that instead of trying to estimate or predict the value in advance, try to shorten the cycle time from development to actual customer feedback, which indicates the value of the software in use. That is, from the SWE perspective, the speed of feedback received from users is the best indicator of the value of the newly created software. This indicates that shortening the feedback cycle would drive the SWE process towards faster reaction on customer value and higher value creation.

Although there exists a common understanding about the key role of a fast customer feedback cycle in linking the SWE process to value creation, the measurement methods and metrics available in literature are positioned either as cost-based SWE methods or as value-oriented metrics with little connection to the engineering process providing little guidance for managing and developing the SWE process (see Table I). Thus, the research question of this paper is, what metrics would guide cycle-time-driven software engineering process development in established organizations?

As the answer is context-dependent, a set of metrics will be needed. This paper aims at filling this gap by proposing a metrics framework enabling adoption of such metrics in a variety of contexts where new features are incrementally added to software.

 ISBN: 978-1-61208-438-1

TABLE I. POSITION OF THIS RESEARCH TO BRIDGE COST-ORIENTED SOFTWARE ENGINEERING (SWE) METRICS AND VALUE-ORIENTED BUSINESS METRICS

	Measurement Domains		
	SWE Metrics	***Research Gap Addressed Here***	***Value Metrics***
Scope (measurement target)	SWE Process	Value Creation Cycle	Customer Value of Offering, Value of Startup
Measured Attribute	Cost, Effort, Quality	Cycle Time	Value for Customer, Value for Enterprise
Examples	Function Points per month, Faults per lines of code		Value in Use, ROI, Lean Analytics

Applying the guidelines of the design science method [21], this research has been initiated based on company needs presented in interviews of Software as a Service (SaaS) development firms in a large industry-driven research program [22], to target an issues with business relevance in firms.

In Section II, we construct the metrics framework artifact based on the analysis and synthesis of previous research literature selected from the perspective of the research question. Following the design science research guidelines, we also demonstrate generalizability of the framework artifact to several contexts by choosing from a variety of metrics to target the specific process development needs. We also propose a simple diagrammatic representation for visualizing some of the metrics values in operational use to pinpoint development tracks requiring attention in an organization with multiple parallel feature-development teams.

In Section III, we evaluate the metrics framework by applying it to the case of a firm developing new features for an existing SaaS service and discuss the impact of the findings on revising the target of the next process improvement actions. In Section IV, we summarize the results, draw the conclusions and propose directions for further research.

II. THE CYCLE-TIME METRICS FRAMEWORK

A. Developing the Framework

The flow of new features through a SWE process can be measured at various points in time with an aim to reduce delay between points to reduce cycle time. The scope of the process measured will impact the attention of the software developing organization. In the narrowest scope, the cycle time measured includes the basic software development cycle while the widest cycle takes into account the customer needs and experience and, thus, matches and even expands the lean startup cycle [20].

In the proposed framework (see right side of Figure 1), the feature life-cycle begins with three planning phase events: 1) a need emerges, 2) a software development organization recognizes the need, and 3) the decision is made to develop the feature. In large established organizations, the identification of feature needs has been excluded from the responsibility of the SWE organization to responsibility of the product marketing organization, while the entrepreneurship-oriented startup community has emphasized the value of including the need identification step as an inherent part in the fast business development cycle of the organization developing the software. Sometimes there is an intentional lag between events 2 and 3 as the decision may be to wait for the right time window (cf. real options [23][24]), or features with higher priority are consuming all resources available.

Continuing from the 3 events that form the beginning of the feature life cycle (above) and for the purposes of measuring the value creation cycle, the main development events included in this framework are 4) development starts, 5) development done, and 6) feature deployed. Use of XP, Scrum and other iterative and incremental development (IID) processes has aimed at reducing the time between events 4 and 5 (or fixing that to 2–4-week cycles). The cycle-time from 4 to 5 is here referred to as the Development cycle (see Figure 1). Moving from packaged software to cloud delivery and SaaS development along with moving from an annual or a six-month software release cycle to continuous integration (CI [25]) and continuous delivery (CD [26]) in development operations (devops [27]) has reduced the interval between 4 and 6.

After the event 6, the traditional software engineering process is often thought to be completed, while many entrepreneurship-oriented approaches, such as Lean Startup [20], go further, starting from building a product to measuring the use of it, which produces data used for learning and for producing ideas for the next development cycle (see left side of Figure 1). That is, building the product based on current ideas is only one of the three main events needed for value creation: build–measure–learn [20]. For considering the business and customer perspectives in this metrics framework for the value creation cycle, we need to expand beyond step 6 to include the use, measuring and learning phases: 7) when the feature gets used, 8) when feedback data is collected to support learning, and 9) when a decision is made based on the feedback. Note that events 8 and 9 resemble events 2 and 3 while not all information from customer needs is collected through measuring the use of the current product. It is also commonly assumed that the time from feature deployed (6) to first use (7) is short, while without measured data this can be an incorrect assumption. There have been cases where almost half of software features were never used [28]. Further, if software quality is high, it can take some time to get feedback, and it may require many uses of the feature before customer sends feedback about problems. Additionally, it can take time for a feature to get sufficient number of uses to allow for a reliable analysis of customer behavior (8). Also, the deployment process of the company can delay the decision to act on the feedback (9).

 ISBN: 978-1-61208-438-1

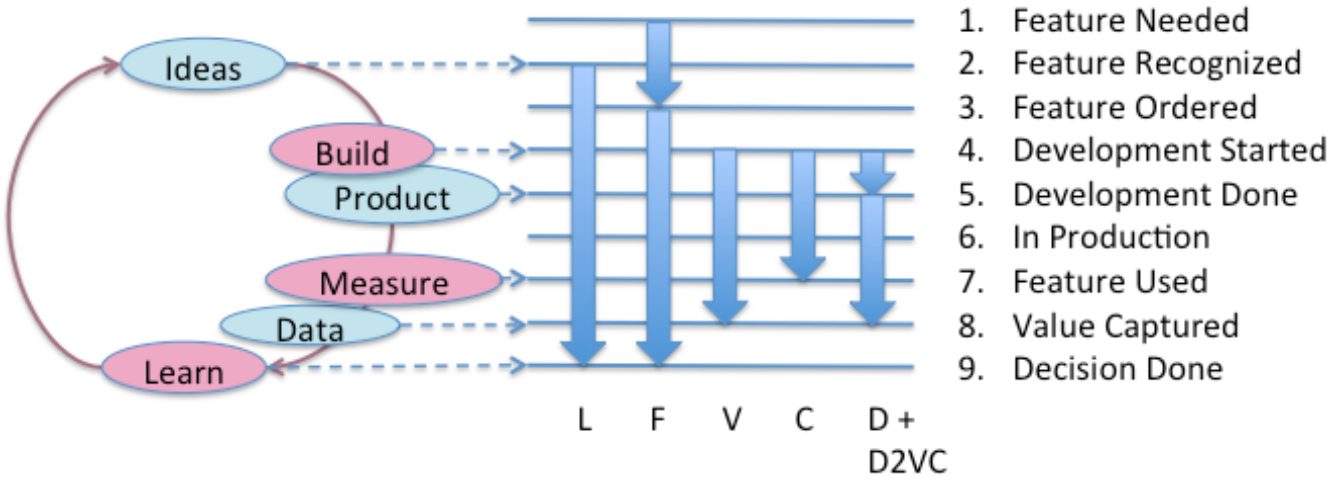


Figure 1. The value-driven metrics framework for driving software engineering cycle-time reduction (on the right), the Lean Startup cycle (on the left) and example cycles, for which cycle time can be used as the metrics driving cycle-time reduction (in the middle).

Figure 1 depicts the proposed framework. On the right side we have the sequence of events identified. On the left side, we have the Lean Startup cycle with horizontal arrows pointing from the phases to related events of the framework. The vertical arrows in the middle represent examples of cycle times that can be used as a target metric for developing SWE process. The cycles in the center are labeled as follows: L = Lean Startup cycle, F = Full cycle including fuzzy front end and full feature development cycle, V = Value cycle from starting the development to value capture, C = Core cycle from development start to first feature use, and finally D+D2VC, where D = Development cycle from start of development to production readiness and D2VC = time from development done to value captured. We emphasize that this list of cycles is not exclusive and new cycle time metrics can be created with this framework on demand for each context.

B. *Changing Process Development Focus through Metrics*

The various cycle-time metrics available in the framework can be used for focusing process development activity to specific process areas based on the need (see Table II). For example, if the basic software development process has been well developed and if some incremental development process, automated testing and continuous integration are applied, it may be useful to shift the attention to continuous deployment. In that case, the metric to be followed can be changed from Development cycle to cycle time between events 4 and 6, from start of development to start of production (see the second line in Table II). Changing the metric will also change the focus of attention and can often result in adjusting the processes, resource allocations or tools used.

TABLE II. EXAMPLE PROCESS DEVELOPMENT TARGETS WHEN USING ALTERNATIVE CYCLE-TIME METRICS

Cycle	Start Event	End Event	Addressed Capabilities	Process Development Focus
D, Development	4: Development Started	5: Development Done	XP, Scrum and other IID processes, automated testing and continuous integration (CI)	Using this cycle-time metrics addresses cycle-time of the basic SW development process
Time to production	4: Development Started	6: In Production	Same as in D, adding continuous deployment (CD) to the measurement scope	Using metrics for this cycle time focuses attention to CD capability
C, Core cycle	4: Development Started	7: Feature Used	Same as previous adding communication (diffusion) to customer base to the scope	Here the focus shifts to integrating customer facing team with development
V, Value cycle	4: Development Started	8: Value Captured	Adding customer analytics and customer feed-back capabilities to the previous scope	Shifts focus to integrating analytics capability to IID+CI+CD capability
Time to Value	4: Development Started	*: Break Even	As Value cycle, but using this metrics assumes that value produced can be evaluated.	As in Value cycle
D2VC	5: Development Done	8: Value Captured	Post-development processes needed to deliver the created value and to get the feedback	Focusing on value cycle capabilities after the basic SW development process.
Fuzzy Front End	1: Feature Needed	3: Feature Ordered	Deep customer understanding (between events 1 and 2) and market understanding (2 to 3)	Measuring capability to find customer needs close to actionable market
...	...	...	...	...

 ISBN: 978-1-61208-438-1

In large organizations, where the product-marketing department is responsible for collecting market requirements and for product launches, the processes crossing product development and product-marketing departments may be problematic. In these cases, choosing the Value cycle, Time to Value or Design done to value captured (D2VC) as a common metric for both of the departments will enforce collaboration between the departments and will likely improve the total value creation capability of the organization, while local metrics within the departments are likely to lead to local optimization leading to non-optimal organizational behavior. It should be noted, that this issue appears mainly in large established organizations rather than in small startup firms, the needs of which the lean startup approach has been developed.

The time to value cycle in Table II ends with the event of reaching the breakeven point, which is marked with an asterisk "*" rather than a number representing a specific ordering in the framework. In some cases a pay-per-use business model provides a basis to determine the break-even point for a feature, while in some cases the break-even point is estimated by qualitative means. A new feature may produce enough value to reach the break-even point when it is published (event 6) or when it is used for the first time (event 7). However, in many contexts this event occurs close to event 8, Value captured, that is, the feature use count is high enough, and sufficient feedback has been received, to ascertain whether the feature was worth the development effort. Based on these examples and the other examples in Table II we can observe, that the choice of applicable metrics is context dependent. Thus there is a need for a framework for metrics, which supports choosing the metric applicable for a specific situation.

C. Depicting Cycle-Time Elements

Depicting the proposed cycle-time metrics makes it easier to decide whether to further develop or even to drop a existing feature and will also help in communicating the cycle-time reduction agenda to software engineers and other parties involved. For this purpose we devised a simple diagrammatic representation presented in Figure 2. In this example, the development starts at point 4 and ends at point 5. The y-coordinate represents the cumulative development time, in line with the cumulative cost for the organization. This linear curve is intentionally simplistic as the focus is on the form of the curve after event 5. In contrast, software engineering oriented representations, such as the Kanban Cumulative Flow Diagram [29], focus on analysis of the development cycle from 4 to 5 and ignore activity after production readiness.

From event 5 on, the horizontal line represents the duration of the waiting time from ready-to-deploy through deployment to first use. The feature is used for the first time in production at event 7. After that the dropping logarithmic curve represents the speed at which feedback has been received. After the second use the curve comes down to half, after the third use to one third of the original, and so on.

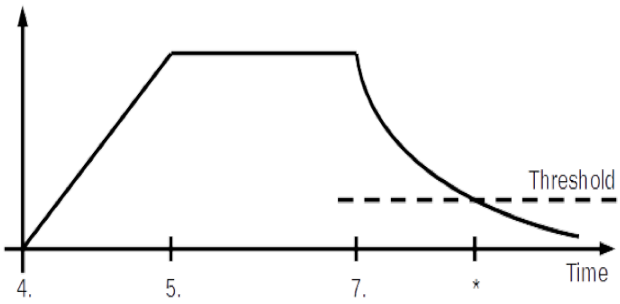


Figure 2. Depiction of the cycle times for feature analysis and process development. The numbers refer to the event number in the framework.

That is, the curve represents development time divided by number of times used. A context-specific target threshold for development time per times used is presented as a dotted line and the time when the curve reaches the threshold is marked with an asterisk "*".

In line with our approach to focus on the cycle times, this graphical representation aims at depicting the cumulative effort invested to the feature during development. There is a risk embedded in this development effort as it has not received feedback from the customers. Thus it is potential waste if customers do not accept the feature. This risk is mitigated along the narrowing gap of the asymptotic curve and the horizontal axis and reaching the threshold indicates that enough customer feedback has been received to ascertain whether it has been worth developing the feature. Event 8, Value captured, is serving this purpose as the event when sufficient user feedback is gained to evaluate the value of the newly developed feature and for adjusting the development plans accordingly, to further develop the feature or to drop it. In addition to guiding value creation, fast feedback from event 8 makes it easier for software developers to fix errors and modify the feature as long as they can still recall the implementation of the feature and have not moved on to new assignments.

Although measuring value is difficult, we would also like to identify the time-to-value cycle, that is the time from starting the development to break even, to the point at which its value to the customer exceeds the development costs. Now we face the challenge that while the cost can be easily measured in terms of time or money, value as a concept is not clearly defined and even if it were it would be hard to measure. We can speculate that the break-even point could be reached already on deployment (for features whose existence provides value even if they are not used, e.g., emergency-situation feature), on first use (when customer finds it), after a certain amount of uses (some use value derived from each), or sometimes a feature can fail to become profitable. Thus, the location of this measurement point cannot, in general, be identified in the sequence of events in the proposed framework, rather it is context dependent.

If we want to measure value, we need to define value. Historically three forms of economic value are the use value, exchange value and price [30]. There are many theoretical divisions of value to support decision making about which

 ISBN: 978-1-61208-438-1

software feature to work on next [2][5][7][8][9][10][11][12][13][14][15][16][24][30][31], but most theories consider the use value to the customer as essential. For the purposes of metrics development the focus will be on customer use value. It is important to note that due to market mechanism, exchange value is less or equal to use value [30]. This means that we could calculate a monetary estimate for the upper bound of the value captured by the software developer, that can be compared with cost. Still, the issue is problematic.

If we assume that there is use value for a feature, and in some cases the use value can be estimated as equal for each use, we would like to measure directly the cost versus benefits ratio: $\frac{development\ costs}{benefits}$. However, as discussed the benefits are challenging to measure and, at worst, we might need a new metric for each feature. This leads us to suggest that we isolate the hard-to-measure part, benefits, by instead measuring the precisely calculable cost per use $\beta = \frac{development\ costs}{times\ used}$ and only if possible compare it to the estimated value for the user, based on a case-by-base estimation method. Next, we will show, using a case study, that reaching events "*" and 8 produce very similar value for process development and feature decision making and that they can be used interchangeably. Thus, time to receive enough feedback is also a good, practical proxy for value produced.

III. METRICS VALIDATION CASE STUDY

A. Target Organization and Service

We evaluated the metrics framework in a mid-sized Finnish software company, Solita Ltd. The case software development team develops a publicly available SaaS (lupapiste.fi) used by citizen applying for a construction permit related to real estate and other structures. This privately operated intermediary service provides a digital alternative to avoid the time-consuming paper-based process of dealing directly with the public authority. This service is used by employees of the licensing authority in the municipalities (about 100 users), the applicants (citizens and companies, about 100 per month), and architects and other consultants (1-2 per application). The software development process metrics were evaluated with the usage data collected from the process flow of five new features of this SaaS service deployed during the observation period, in mid-2014.

The service has a single page front-end that connects through a RESTful API to its back-end. Each call to the API is recorded on the production log files with a time stamp. We mined and analyzed the log entries together with the development data captured by the version control system. In this case, we chose features that introduced a new service to the API and were thus possible to trace automatically with a simple script that queried the monitoring system automatically. Some manual work was needed to find the features that introduced a new API, but automation of this work is also possible.

B. Results from Applying the Metrics to Sample Features

From the recorded event time stamps we calculated three metrics values for the case features. Development cycle (D) from start (4) to done (5) in working days. Lag to production from done (5) to deployed (6) in calendar days. And finally, D2VC, time from development done (5) to value captured (8). In this context the target company estimated that enough feedback data was collected for learning when the feature was used four times per each day spent on development, which gave the context-specific definition for the value capture event (8). Table III presents the data that is depicted in Figure 3. To enable comparison, all the features are shifted in the time axis to have event 4 (start of development) at day 0. In a daily use, an alternative depiction can show the timeline representing the history of all features to current point of time from which it is easy to identify development peaks and, more specifically, to notice the curves that remain high after the peak which indicates a demand for action. Either a feature has not been deployed and promoted well for the users or there is no user need for the feature.

C. Case Analysis and Discussion

From Table III we can see that for these five features the average of development effort needed to implement and test the features was about eight working days. When the development was done, on an average 12 calendar days was spent on waiting for deployment of the feature to the production environment. We can also observe that the features with lower priority (F1647 Unsubscribe and F1332 Note) have almost double the lag to production compared to the other features.

TABLE III. DESCRIPTION OF THE SAMPLE FEATURES, THEIR PRIORITY, DEVELOPMENT TIME (IN WORKING DAYS), LAG TO DEPLOYMENT (IN CALENDAR DAYS) AND DEVELOPMENT TO VALUE CAPTURED (D2VC; IN CALENDAR DAYS)

Feature id	Priority	Development (days)	Lag to deployment (days)	D2VC (days)	Description of the feature
F1332 Note	2	10	24	24	Authority user can add a textual note that other users cannot see.
F1498 Attachment	4	9	10	N/A	Applicant user can set the target of an uploaded attachment.
F1507 Validate	4	10	1	49	System validates the form prior the user sends the application.
F1537 Sign	4	7	11	15	Authority user can require an applicant to sign a verdict.
F1647 Unsubscribe	3	2	15	28	Authority user can unsubscribe emails.
Average		**8**	**12**	**29**	

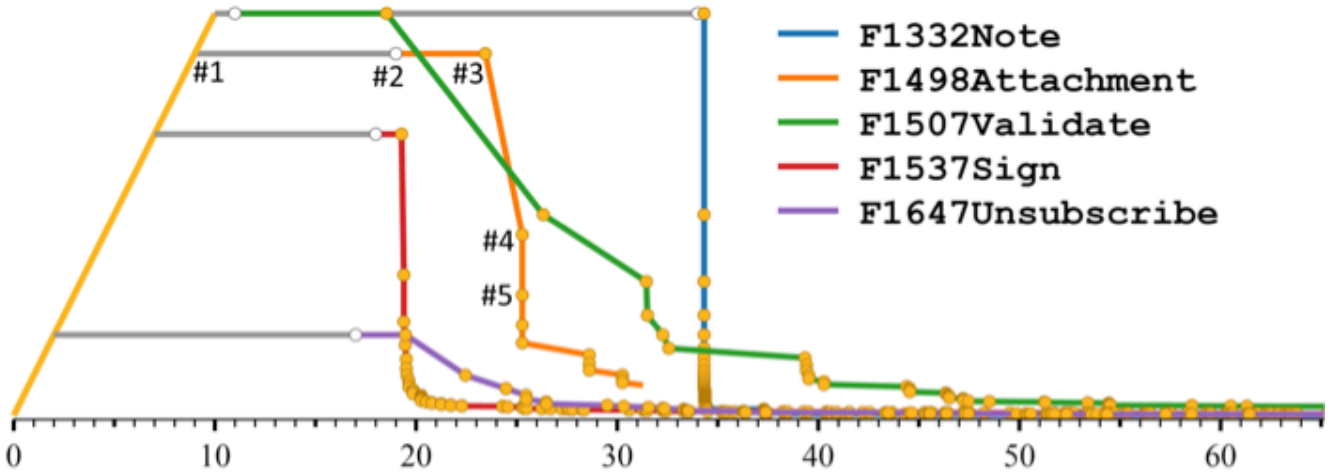


Figure 3. Depiction of the cycle-times of the five features. Development working days share the rising line starting from (event 5) and end in event 6 (start of the gray horizontal line), deployment (7, white dots in the right end of the gray part of the horizontal lines) and usage (yellow dots). To enable comparison, all the features are synced to have event 5 (start of development) at day 0.

The average time from completion of development to value capture is 29 days (this does not include feature F1498 Attachment, which did not reached the number of uses needed for the threshold). From the depiction in Figure 3, we can also see that this feature is no longer used. This feedback triggers the discussion on the reasons for the discontinuation of use of the feature to determine if there is a need to improve it or remove it from the service. When the target is to minimize the cycle times, minimizing the lag from production readiness to deployment (from event 5 to 6) and the means to increase the use of new features are clearly the places where major improvement can be reached much easier than from reducing average development time. By plotting the events in this way, it is easy to identify the places where changes can be made as well as to communicate the need with the development teams.

The results triggered also a discussion on the release practices of the firm. From the service use statistics it is possible to see that the service is heavily used from Monday to Thursday, less on Friday and very little during weekends. Thus it is likely that features released on Mondays will get used sooner than the ones released on Fridays, which provides the additional benefit that the feedback from users (8) would reach the developers when they still recall the software they were working on. Even more profound than the weekly cycle is a similar variation related to the vacation seasons. Deploying new features just prior to vacations will have negative impact on the Value cycle, as described above.

IV. Summary, Conclusions and Further Research

The feedback from practitioners suggests that the current literature lacks metrics that could be used for directing a software development organization from the business perspective to enhance effective value creation and value capture. Although the Lean Startup Methodology proposes to develop the software via the build–measure–learn cycle, we seem to lack the means to measure the value that the delivered software creates. Also the researchers have observed this problem and conclude [7], that the current theories of value do not present a simple way of assessing customer-perceived value. Although companies put a great amount of effort into increasing customers' perceived value in the product development process, determining how and when value is added is still a challenge even in marketing and management sciences [7]. Previous literature on XP, Scrum, lean startup and related approaches has indicated that in the context of SaaS services, delivering new versions of a service to the customer, collecting the usage data and making further decisions based on the data provides the most promising path for the software vendor to understand customer-perceived value. Agile methods have been shown to enable shorter lead times and a stronger focus on customer needs [32].

Shortening the cycle times provides increased flexibility maintaining options to change development direction with speed [20][22] as well as other business benefits for software service firms. This encouraged us to search for metrics that help software firms in the process development towards shorter cycles. On this basis, we formulated the research question as, what metrics would guide the cycle-time-driven software engineering process development in established organizations?

As the proposed solution, we adopted and extended the lean startup [20] value creation cycle and constructed a framework for metrics based on the times between main observable events within the cycle, all the way through to receiving and analyzing user feedback. This focuses attention on fast execution of the value creation within the user feedback cycle. That is, we are not trying to measure value of the results of the cycle, such as the value of the product produced or the value of the startup or progress of the startup in creating the offering, as in lean analytics [12].

 ISBN: 978-1-61208-438-1

By finding the measurable values from within the value creation cycle, the cycle-time metrics framework aims at bridging the gap between cost-oriented SWE metrics and value-oriented business metrics. Cycle-time reduction serves as the intermediary of increased value creation guiding software feature development and software process development. The metrics measure the calendar time between the key events. The first three events are related to feature need identification and the business decision to implement the specific feature (event 3). The core events following this decision are start of the development (4), the feature is ready for deployment (5), the feature is deployed (released, 6), and first use of the feature by a customer (7). These events are followed by feedback related events, the feature feedback data has been collected and analyzed (8) and a decision is made based on the feedback (9). The time intervals between the core events (4-7) are of most interest for the engineering while the other events (1-3 and 8-9) relate to the customer-perceived value analysis of the feature. We also provided examples on how changing the measurement cycle directs the process development to new process areas.

Our empirical focus was at the level of features being added to an existing SaaS offering. In the empirical part, the times between the events in the core cycle were measured for five new features in the development processes of an independent software vendor's SaaS service. The results showed that the core metrics were able to capture and bring up useful characteristics of the business process that triggered both a "drop vs. develop feature" discussion (for feature F1498) and a number of process development discussions. In these five feature development cases the average development time was shorter than the waiting time for the feature to be released. This has negative impact to the efficiency of fixing potential problems emerging during the first uses of the feature by first users, as the developers have already oriented towards another assignment. The detection of the delay of feature releases lead to a further analysis of the vendor's release practices in general and prompted quick improvements to their process.

Although the results from the empirical part showed that the metrics are useful in practice, there are still several avenues of further research that we wish to explore. The empirical part used data from the engineering system and customer feedback data to identify the core events. This seems to be a useful starting point and the firm in our case study would like to extend the collection of data to cover as many of the nine events as possible and as automatically as possible. The time from release readiness to analyzed customer feedback seems to be a particularly useful measurement of deployment performance.

In general, collecting the data can and should be automated using engineering information systems to the extent possible (events 1 and 8 cannot be detected automatically). For the other events, we propose collecting and depicting the data graphically in real-time status displays providing an overview of the development activities for business and engineering management. As we can observe from the empirical case, the results are useful both for focusing process development activities and for making business decisions regarding which features will be developed further, which will be used as they are, and which features will be removed from the service. This way the simplified depiction can provide transparency between the business and the development organization. Thus we encourage further empirical work on the automation of data collection and its depiction based on events identified in the framework.

In startups the result of value creation cycle can be analyzed in the context of the evolution of the enterprise [12]. In context of established feature development processes, this framework adopted the approach of using only cycle times between events as the metrics within the value creation cycle. This is due to limited applicability of suitable previous research results for real-time customer-perceived value analysis beyond A/B testing and similar tools that can be used between events 7 and 8. Although cycle time metrics seems to provide high added value to focus process development in connecting software development with customer value, investigating the value capture events 8 and "*" further is needed. Finding an easy to apply means for estimating the perceived user benefits would enable various new developments supporting the operative business development of a software engineering team.

ACKNOWLEDGMENT

This work was supported by TEKES as part of the Need for Speed (N4S) Program of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business).

REFERENCES

[1] B. W. Boehm, "Value-based software engineering: Overview and agenda," in Value-based software engineering, Springer Berlin Heidelberg, 2006, pp. 3-14

[2] M. Khurum, T. Gorschek, M. Wilson, "The software value map - an exhaustive collection of value aspects for the development of software intensive products," Journal of software: evolution and process. Wiley, 2012, 711-741.

[3] H. Erdogmus, J. Favaro, W. Strigel, "Return on investment," IEEE Software 3(21), 2004, pp. 18–22.

[4] K. Conboy, "Agility from first principles: reconstructing the concept of agility in information systems development," Information Systems Research, 20(3), 2009, pp. 329-354.

[5] S. Barney, A. Aurum, C. Wohlin, "A product management challenge: Creating software product value through requirements selection," Journal of Systems Architecture, 54(6), 2008, pp. 576-593.

[6] A. Fabijan, H, Holström Olsson, J. Bosh, "Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review," in Software Business. Springer International Publishing, 2015, pp. 139-153.

[7] J. Gordijn, and J.M. Akkermans, "Value-based requirements engineering: exploring innovative e-commerce ideas," Requirements Engineering 8(2), 2003, pp. 114-134.

[8] M. Rönkkö, C. Frûhwirth, S. Biffl, "Integrating Value and Utility Concepts into a Value Decomposition Model for

 ISBN: 978-1-61208-438-1

Value-Based Software Engineering," PROFES 2009, Springer-Verlag, LNBIP 32, 2009, pp. 362–374.

[9] R.B. Woodruff, and F.S. Gardial, Know your customer: New approaches to customer value and satisfaction. Cambridge, MA, Blackwell, 1996.

[10] C. Grönroos, "Value-driven relational marketing: from products to resources and competencies," Journal of Marketing Management 13(5), 1997, pp. 407–419.

[11] T. Woodall, "Conceptualising 'value for the customer': An attributional, structural, and dispositional analysis," Academy of Marketing Science Review, no. 12, 2003, pp. 1526–1749.

[12] A. Croll, and B. Yoskovitz, Lean Analytics: Use Data to Build a Better Startup Faste,. O'Reilly Media, Inc. 2013.

[13] P. Tyrväinen, and J. Selin, "How to sell SaaS: a model for main factors of marketing and selling software-as-a-service," in: Software Business, Springer, Berlin Heidelberg, 2011, pp. 2-16.

[14] V. Mandić, V. Basili, L. Harjumaa, M. Oivo, J. Markkula, "Utilizing GQM+ Strategies for business value analysis: An approach for evaluating business goals," The 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ACM, 2010.

[15] M. Saarikallio, and P. Tyrväinen, "Following the Money: Revenue Stream Constituents in Case of Within-firm Variation," in: Software Business. Springer International Publishing, 2014, pp. 88-99.

[16] J. Bosch, "Building products as innovation experiment systems," in: Software Business, Springer, Berlin Heidelberg, 2012, pp. 27-39.

[17] K. Schwaber, and M. Beedle, Agile Software Development with SCRUM, Prentice Hall, 2002.

[18] K. Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999.

[19] A. Cockburn, Agile Software Development, 1st edition, 256 p. Addison-Wesley Professional, December 2001.

[20] E. Ries, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Publishing Group, 2011.

[21] A.R. Hevner, S.T. March, J. Park, S. Rami, "Design Science in Information Systems Research," MIS Quarterly, Vol. 28, No. 1, 2004, pp. 75-105.

[22] J. Järvinen, T. Huomo, T. Mikkonen, P. Tyrväinen, "From Agile Software Development to Mercury Business," in: Software Business. Towards Continuous Value Delivery, Springer Berlin Heidelberg, LNIB, vol. 182, 2014, pp 58-71.

[23] H. Erdogmus, and J. Favaro, "Keep your options open: Extreme programming and the economics of flexibility," in Giancario Succi, James Donovan Wells and Laurie Williams," Extreme Programming Perspectives", Addison Wesley, 2002.

[24] M. Brydon, "Evaluating strategic options using decision-theoretic planning," Information Technology and Management 7, 2006, pp. 35–49.

[25] M. Fowler, Continuous Integration, 2006. http://martinfowler.com/articles/continuousIntegration.html retrieved: Septmeber, 2015.

[26] J. Humble, and D. Farley, Continuous delivery: reliable software releases through build, test, and deployment automation, Pearson Education, Jul 27, 2010.

[27] P. Debois, "Devops: A software revolution in the making," Cutter IT Journal, vol. 24, no. 8, August, 2011.

[28] J. Johanson, Standish Group Study, presenation at XP2002.

[29] K. Petersen, and C. Wohlin. "Measuring the flow in lean software development." Software: Practice and experience, vol. 41, no. 9, 2011, pp. 975-996.

[30] J.S.Mill, Principles of political economy, 1848, abr. ed., J.L.Laughlin, 1885.

[31] M. Cohn, Agile estimating and planning. Pearson Education Inc. 2006.

[32] M. Poppendieck and M.A. Cusumano, "Lean software development: A tutorial," Software, IEEE, vol. 29, no. 5, 2012, pp. 26–32.

 ISBN: 978-1-61208-438-1

A Context-Driven Approach for Guiding Agile Adoption: The AMQuICk Framework

Hajer Ayed, Benoît Vanderose and Naji Habra
PReCISE Research Center
Faculty of Computer Science, UNamur
Rue Grandgagnage 21, B-5000 Namur, Belgium
emails: {hajer.ayed, benoit.vanderose, naji.habra}@unamur.be

***Abstract*—Regarding the proven benefits of agile software development, more and more practitioners are becoming interested in agile methods and have to deal with the complexity and costs of the adoption process. In this context, agile experts argue that prior to any agile method or practice adoption, its relevance to the organization and team should be evaluated to avoid unnecessary implementation efforts and resources. The goal of this research is to investigate a context-driven approach for guiding agile methods adoption: starting from the characterization of the context properties, the approach helps to identify relevant practices and to recommend process customization using adequate rules. The focus in this paper will be on the agile context characterization using relevant, reusable and measurable elements structured in a context metamodel. A purposely simple instantiation is proposed to illustrate how customization rules would be inferred from the context characterization.**

***Index Terms*—agile software development; software process customization; agile context; agile practice selection.**

I. Introduction

Even though the benefits of agile methods have been proved by successful implementations and experiences, the complexity of adopting them is high and requires lots of effort: upper management sponsorship, customer involvement, team empowerment, traditional organizational silos replacement with cross-functional teams, deals with egos and resistance to change, business model arrangement, etc.

To take advantage of the agility benefits and to overcome these common issues, experts and practitioners highlight the necessity to properly adapt practices, deliverables, activities and any other process aspect to avoid unnecessary implementation costs and efforts and to better accommodate the team's specific context and needs.

The agile literature, as explained by Dybå et al. [1], provide a broad picture of adaptation experiences and successful agile implementation but most of them are hardly reusable because they lack of structuring and are often based on experts knowledge and intuitive reasoning: the adaptation decisions are neither documented nor structured nor automated (see section II).

The goal of the *AMQuICk* framework [2][3] is to provide methods and tools to guide the adaptation of agile methods in a more objective, structured and (at least partially) automated way. To that end, it is necessary to record and formalize the intuitive knowledge of agile experts regarding the adaptation of methods to specific contexts so that decision-making may be systematized.

A key to success in this endeavor lies in the exploitation of a formalized and measured representation of the context of an ongoing development process. Given an objective model of the context (including measured attributes), the identification of relevant process elements to recommend to the team maybe more easily exploited by formalized recommendation rules (paving the way towards a complete expert system).

This paper introduces an approach to context-modeling designed to be exploited in such a way and demonstrates how an agile context can be instantiated using relevant measurable elements in order to infer customization rules. The main questions underlying the approach are therefore: (1) how can we model and compose agile processes using reusable components?, (2) what defines an agile software development context and how to model it?, (3) how to retrieve relevant components regarding the context at hand?.

The remainder of this paper focus on context modeling challenges and is structured as follows: Section II presents the existing approaches and context models to guide the customization process. Section III-A presents an overview of the framework that we propose. Sections III-B and III-C provide details on the process specification and context modeling. Section III-D refers to the formalization of the process engineering interpretation rules. An example is presented in Section IV to illustrate the context metamodel instantiation and how customization rules could be inferred from the context characterization. Finally, Section V presents closing comments and future work.

II. Related work

A. Agile Customization

Even though the literature abounds with valuable agile methods tailoring experiences reports [1], most of them are difficult to exploit, because too narrowly linked to a specific situation and often based on experts' knowledge and intuitive reasoning: neither documented nor structured nor tool-supported.

There exist structured approaches that provide practical road-maps to facilitate and guide through the implementation and tailoring process [4][5] but they definitely lack automation: most of them are just documents with guidelines and repeatable steps to follow for effective agile methods implementation. Moreover, the problem with these approaches is that each of them proposes a solution based on only few and prefixed factors influencing the implementation. For example, Cockburn [6] proposes to choose the agile methods among the Crystal family methods according to the *number of people involved* and *criticality* criteria .

 ISBN: 978-1-61208-438-1

Other approaches, such as Mnkandla [7], propose a toolbox for only practices selection and not other process aspects. Moreover, the linkage with project context is only allowed through a predefined methodology selection matrix and project taxonomy matrix. This kind of matrix synthesizes some experts' knowledge therefore preventing any possibility of extension.

Finally, more formalized and tool-assisted approaches, such as Mikulėnas et al. [8], aim to support agile methods adaptation by providing users with rich practices composition mechanisms (e.g., merging, coupling, etc.). However, the choice of suitable practices is only based on the user appreciation: the adaptation decisions are not assisted or derived from context attributes.

B. Agile Context Defined

The software development context refers to all the influential circumstances and variables that affect the work environment of all stakeholders involved in the project life-cycle, e.g.,: *market uncertainty, budget constraints, application domain, project criticality, project duration, team size, familiarity with the involved technology, etc.*

Although the term context has an intuitive meaning for agile practitioners, it's hard to formalize the relevant context variables to support software process adjustments.

Several contextual models to guide the adoption and adaptation of agile software development practices can be found in the literature.

Cockbrun et al. in the crystal family of processes [6] define different processes based on *Product Size*, *Criticality*, and *Skills*.

Boehm et al. [9] define a home ground of agile vs. plan-driven as associated to five critical factors namely, *Product Size*, *Criticality*, *Dynamism* (i.e., requirements change rate), *Personnel* (i.e., level of method understanding [10]) and *Culture* (of the team: thriving on chaos or on order).

Kuchten [11] defines 2 sets of factors that make up the context: factors that apply at the level of the whole organization, and factors that apply at the level of the project. The organization-level factors do influence heavily the project-level factors which should drive the process to adopt. The organization level factors are defined as: *Business domain, Number of instances, Maturity of the Organization, Level of Innovation* and *Culture*. Project-level context factors are: *Size*, *Stable Architecture*, *Business Model* (contracting, money flow, etc.), *Team Distribution*, *Rate of Change*, *Age of System*, *Criticality* and *Governance* (management style).

S. W. Ambler [12], in the Agile Scaling Models (ASM) framework, defines a range of 8 scaling factors for effective adoption and tailoring of agile strategies: *Team size, Geographical distribution, Regulatory compliance, Domain complexity, Organizational distribution, Technical complexity, Organizational complexity and Enterprise discipline*.

Even though the context models reported above have been defined for different purposes (i.e., Crystal family of methods configuration, defining agile vs. plan-driven home grounds, practices adoption guidance and scaling agility to larger scopes), they seem to be more or less similar with only minor variations. They are all composed of context *"factors"* or *"dimensions"* at the higher levels refined in a set of *"properties"* or *"attributes"* at the lower levels.

Based on this observation, our target was to find a way to abstract context modeling in a common paradigm, so that agile process engineers or facilitators (or any other equivalent role) can design their own profile to contextualize process components depending on their own perception. Indeed, the set of relevant context elements to support the software process adjustments is potentially different from an organization to another.

The approach investigated in this paper is an attempt to address the issues mentioned above. The following sections provide an overview of the essential set of components required for context-driven adaptation.

III. AMQuICk Framework

A. Overview

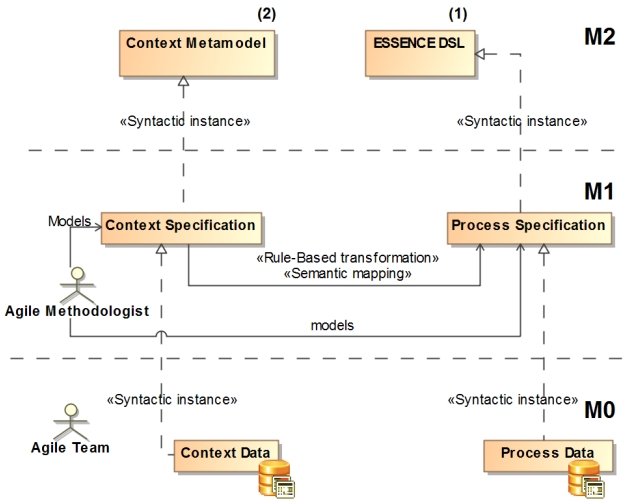


Figure 1: AMQuICk Basic Elements

In order to support the long-term vision of assisted adaptation of agile development processes, it is crucial to complement existing agile tailoring approaches with more objective and systematic guidelines. As explained in Section II, the context appears as a missing link in the formalization process regarding the tailoring of agile methods. The approach we propose is therefore aiming at better formalization of the context, so that it can be exploited further in the process tailoring part of the approach. Practically, the approach relies on a formal (and rule-based) mapping between process models and the related context models.

As illustrated in Figure 1, our approach is model-driven and complies to the Meta-Object Facility (MOF) architecture [13]. At the M2-level lie the two required metamodels: the first dedicated to the context specification, the second to process

 ISBN: 978-1-61208-438-1

modeling. The former is specifically designed for our purpose while the latter takes advantage of the preexisting Essence DSL to define the abstract syntax of agile processes (see Sections III-B and III-C for further details).

While the M0-level is concerned with actual collected data regarding the context and the process, the M1-level is the cornerstone of the approach. At this level, two syntactic instances of the metamodels may be compared and mapped against each other from a semantic point of view. In other words, at this level, a specific piece of context may call for a specific piece of process. This relationship between context and process must be guaranteed by rules derived from the body of knowledge of agile experts referred to in Section III-D.

Implementing this approach is key regarding the elicitation of objective decision-making elements that are needed to guide the evolution and decide which process adjustments to include at the right time. The components illustrated in Figure 1 form the basis of a rule-based system so that the experts can define the crucial context features that influence process adaptation and the practitioners simply enter some information about the project context (by instantiation of the latter features) and get an indication of the most appropriate adaptations for that project.

B. Process Modeling

As explained in Section I, the need for a better flexibility of software engineering motivates the construction of tailored processes to the situation at hand. This discipline is known as Situational Method Engineering (SME).

The kernel of SME consists in composing contextual methods by reusing structured "components" of existing methods. Various techniques can be used including Metamodeling, Domain Specific Languages (DSL), Ontologies, etc.

For the needs of the previously described approach we investigated and compared some of them among which, ISO/IEC 24744 metamodel [14], Software Process Engineering Metamodel (SPEM 2.0) [15] and the recently published DSL Essence 1.0 [16].

The DSL Essence 1.0 (see Figure1) is a Kernel And Language For Software Engineering Methods adopted by the OMG. It was chosen because of its intuitiveness (graphical notation, different level of abstraction: static and operational view), usability at the team level, extensibility and finally because the DSL has been developed among an active initiative which aims to develop a software engineering kernel for both agile and waterfall ways [17].

However, the Essence DSL has to be extended in order to allow for a more structured definition of context-related elements into the process modeling. As explained in [18], quality-related information may be taken into account in order to provide more objective (or a least more structured) elements of context.

C. Context Modeling

In this section, we investigate the second problem introduced in Section I, i.e., what attributes the agile software development context encompass and how it can be captured?

Figure 2 depicts the designed metamodel for context specification. It defines all the concepts (and relationships between them) that may be used in the definition of a context model. The core elements for characterizing an agile context are:

- *"Context Dimension"*: includes the high-level key-concepts of software engineering characterizing the context. Possible instantiations are project, organization, team, endeavor, customer, solution, etc. These instantiations highly depends on the tacit knowledge of agile experts.
- *"Context Property"*: the set of variables underlying a context dimension. For example, the "requirements dimension" may be characterized by the "requirements change" property.

The metamodel also includes concepts designed to describe in details the measurable entities of the context and the nature of the measures themselves (see Figure 2). This subset of the metamodel results from the conceptual alignment of various related works dedicated to define a unified terminology of software measurement metamodels specifically ISO/IEC SQuaRE [19] and the Model-Centric Quality Assessment (MoCQA) [18].

As a result, the metamodel includes all the concepts required to define a well-formed and coherent measurement method: it relies on the notion of measurable entity for which a given measurable attribute has to be mapped to a value (i.e., the measure itself). In the context metamodel, those concepts are replaced with the *"ContextDimension"* (measurable entity) and the *"ContextProperty"* (measurable attribute).

In order to be conceptually correct and allow for the right operation and comparison, the measure has to be identified by a series of variables (i.e., type of value, unit, and scale) that indicates how the sheer value must be understood, compared to other measurement values and interpreted in fine.

The metamodel also emphasizes the difference between sheer measurement values and so-called indicators. Indicators are the key to the approach since they allow bridging the gap between objectified (through measurement) context elements and derived process elements. As explained in [18], indicators are only as useful as their interpretation rules.

In a more traditional quality assessment context, the interpretation associated to a given indicator determines the action to be undertaken in the next steps of the development. Building upon this notion, the *AMQuICk* approach proposes to link the interpretation to process engineering rules (thanks to *"CustomizationRule"*) so that the interpretation of the indicator impacts directly the way the process is to be refined.

D. Rule-Based Process Engineering

Every time an organization is going to develop a project in an agile way, a context profile has to be instantiated using the context metamodel described in Section III-C.

However, a standalone context model has no meaning if not linked with adequate customization abilities. Indeed, the instantiated model only defines the structural properties of

 ISBN: 978-1-61208-438-1

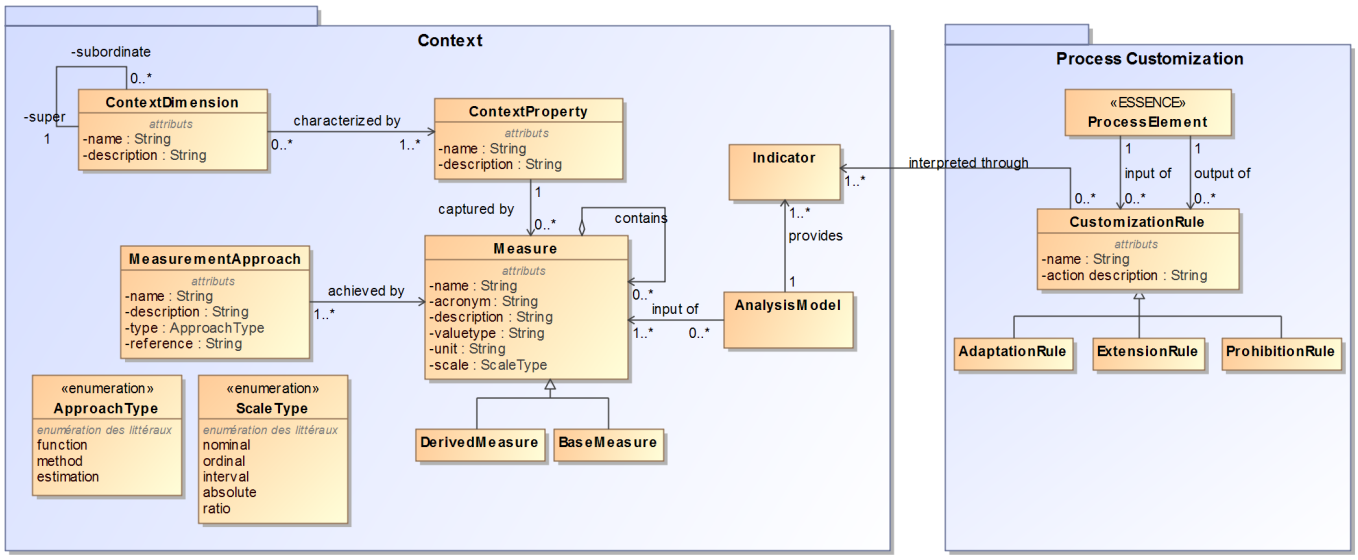


Figure 2: Context Metamodel

the context. The behavioral mechanisms regarding the process engineering have to be elicited.

To do so, the body of knowledge required in the engineering process has to be gathered, i.e., the information about typical project contexts involved in the previous developments of the organization, the information sources of past agile experiences, the tacit knowledge of experts (or experienced people), the previous process configurations and adaptations, the tailoring guidelines (if they exist), etc.

Then, this information has to be structured properly in order to support process engineering decision-making. The relationship between the context and process can be guaranteed by transformation rules. The package *"Process Customization"* of Figure 2 describes the abstract syntax of the interpretation rules engine. *"CustomizationRule"* is associated to:

- a context *"Indicator"* which determines the action or event to be undertaken,
- an input process element (*"LanguageElement"*),
- an output process element (*"LanguageElement"*),

The *"CustomizationRule"* may be of 3 types: an adaptation rule (e.g., iteration length adaptation, start integration earlier, write acceptance criteria before implementation, etc.), an extension rule (e.g., lean value stream map integration, start iterations with model storming sessions, etc.) or a prohibition rule (e.g., collective code ownership and pair-programming are inapplicable in some contexts).

This part of the approach is to be further developed in the future. At this stage of the research, the process customization subset acts as a placeholder that will be refined in the future so that the customization rule generation will be automated, which would not be feasible with the actual formalism. Language elements from the Essence 1.0 DSL in particular "Extension Element" and "MergeResolution" can be reused to compose the output process model [16][17].

IV. Illustration

In order to illustrate an application of the *AMQuICk* contextualizing approach, we propose a small example in Figure 3. The example models the context features used to detect the lack of customer involvement and an adequate adaptation rule.

The customer is here defined as a context dimension characterized by the *customer involvement* context property. This property may be measured in different ways:

- *commitment time*: base measure which refers to the effective time of collaboration between the customer and the development team. This measure is of type *ratio* and is expressed in *datetime*,
- *physical proximity*: base measure which refers to the geographical distance and is expressed in *km*,
- *communication channel*: base measure which refers to the more frequent channel used in the communication between the customer and the team. The measure is of type *nominal* with a range of possible values, i.e., face-to-face, video, phone, mail or documents.

The association between these measures in a relevant analysis model (*involvement analysis* model) provides two indicators of whether the customer involvement is satisfactory or not. In the case of lack of customer involvement detection, a possible process engineering rule to be undertaken is to extend the process with the *[Customer Proxy]* practice [20] (*Enhance Customer Involvement* rule).

The provided illustration in Figure 2 represents a purposely simple context model. Although the ultimate goal of the context model is not to be a visual representation, it is meant to illustrate the kind of objective and measurable information that could be captured by such a model. Similarly, the customization rule provided herein is not meant to be used in such a simplistic way. At this stage of the research, the customization

 ISBN: 978-1-61208-438-1

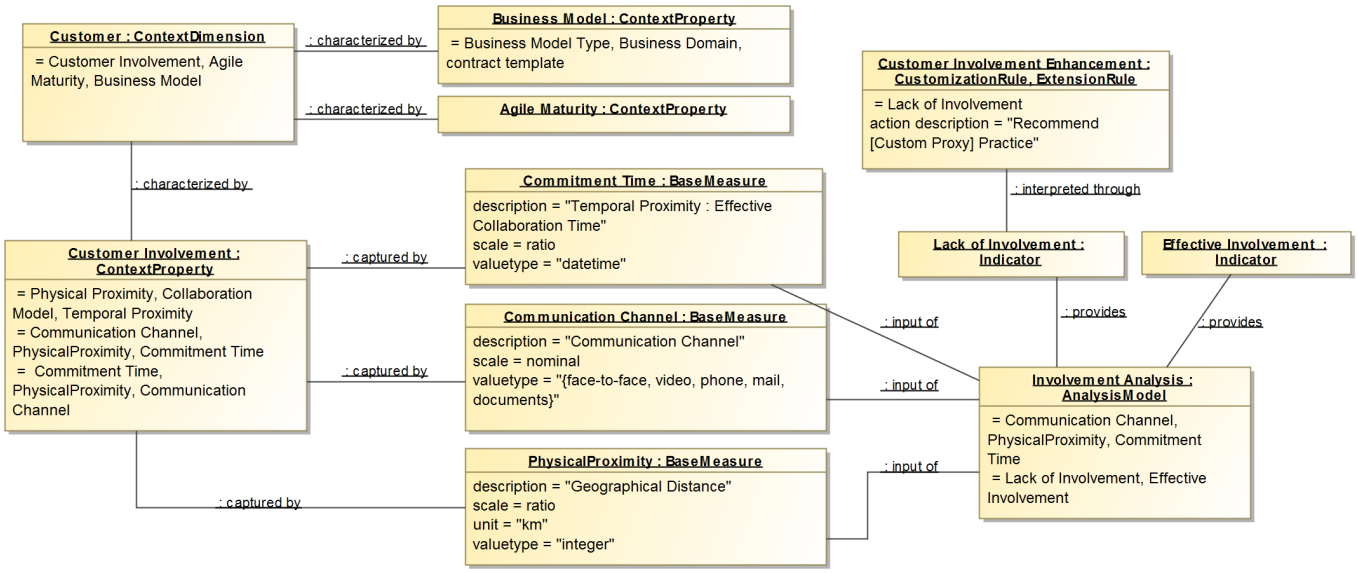


Figure 3: Illustration of a Context Model

rule acts as a placeholder that will be refined in the future so that it can be automated (which would not be feasible with a simple textual representation).

V. Conclusion and Future Work

The approach we propose in this paper aims at supporting decision making regarding agile process (or even any process) evolution in a contextualized way. It relies on an explicit modeling of relevant context-related aspects as well as the use of measurement-based elements to provide objective hints regarding the required process adaptation to undertake. At this stage, the approach focus on the conceptual steps, that is, defining relevant metamodels and conceptual elements.

These conceptual tools are required in order to make the modeling of processes, context elements and process engineering rules possible and coherent. Efforts regarding this conceptual level is still required. For instance, the opportunity to further align the proposed context metamodel with the process metamodel in order to limit the conceptual complexity should be investigated (e.g., the notion of *"Alpha"* from Essence 1.0 and the notion of *"Context Dimension"* proposed in our approach are similar and may be associated).

However, the main added value of this conceptual level lies in the fact that it lays the foundation of a tool-supported methodology. Indeed, the ultimate goal of the approach is to provide an assisted methodology that relies on an expert system. The conceptual approach described in this paper is expected to enable the design of such a system. Indeed, the approach assumes that by exploiting the available agile experiences feedback, we would be able to extract significant and useful knowledge for enhancing the decision-making ability of agile professionals when composing a suitable process.

By exploiting these available agile experiences feedback and linking them to context models, a knowledge database could be populated and enhance the decision-making abilities of the development team. Rules would not only be created by agile experts, they would also be generated through an inference engine.

In turn, this knowledge base could provide the basis of a community-based approach, where continuous feedback, cross-referenced with basic inferences rules, provides an ever improving support for process related decision-making.

References

[1] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9, pp. 833–859, 2008.

[2] H. Ayed, N. Habra, and B. Vanderose, "AM-QuICk: a measurement-based framework for agile methods customisation," in *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*. IEEE, 2013, pp. 71–80.

[3] H. Ayed, B. Vanderose, and N. Habra, "Supported approach for agile methods adaptation: an adoption study," in *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*. ACM, 2014, pp. 36–41.

[4] K. Conboy and B. Fitzgerald, "Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 20, no. 1, p. 2, 2010.

[5] I. Attarzadeh and S. H. Ow, "New direction in project management success: Base on smart methodology selection," in *International Symposium on Information Technology (ITSIM)*, vol. 1. IEEE, 2008, pp. 1–9.

[6] A. Cockburn, *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004.

[7] E. Mnkandla, "A selection framework for agile methodology practices: A family of methodologies approach," Ph.D. dissertation, Faculty of Engineering and the Built Environment, University of The Witwatersrand, 2008.

[8] G. Mikulėnas, R. Butleris, and L. Nemuraitė, "An appraoch for the metamodel of the framework for a partial agile method adaptation," *Information Technology And Control*, vol. 40, no. 1, pp. 71–82, 2011.

[9] B. Boehm and R. Turner, *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley Professional, 2003.

 ISBN: 978-1-61208-438-1

[10] A. Cockburn, "Selecting a project's methodology," *IEEE Software*, vol. 17, no. 4, pp. 64–71, 2000.
[11] P. Kruchten, "Contextualizing agile software development," *Journal of Software: Evolution and Process*, vol. 25, no. 4, pp. 351–361, 2013.
[12] S. W. Ambler, "The agile scaling model (asm) : Adapting agile methods for complex environments," IBM, Tech. Rep., December 2009.
[13] *ISO/IEC 19502:2005 Information technology - Meta Object Facility (MOF)*, International Organization for Standardization and International Electrotechnical Commission Std., 2005.
[14] ISO, *ISO/IEC 24744: Metamodel for Development Methodologies*, International Organization for Standardisation (ISO) Std., 2007.
[15] *SPEM (version 2.0): Software & Systems Process Engineering Metamodel Specification*, Object Management Group (OMG) Std., 2008.
[16] *Essence (version 1.0): Kernel and Language for Software Engineering Methods*, Online at : http://www.omg.org/spec/Essence/1.0, Object Management Group (OMG) Std., November 2014.
[17] I. Jacobson, P.-W. Ng, P. McMahon, I. Spence, and S. Lidman, "The essence of software engineering: the semat kernel," *Queue*, vol. 10, no. 10, p. 40, 2012.
[18] B. Vanderose, "Supporting a model-driven and iterative quality assessment methodology: The MoCQA framework," Ph.D. dissertation, Ph. D. dissertation, University of Namur, 2012.
[19] W. Suryn, A. Abran, and A. April, "ISO/IEC SQuaRE: The second generation of standards for software product quality," in *7th IASTED International Conference on Software Engineering and Applications*, 2003.
[20] A. Alliance, "Agile alliance guide to agile practices," *Online at: http://www.guide.agilealliance.org/. Last accessed 02/10/2015*.

 ISBN: 978-1-61208-438-1

Kanban in Industrial Engineering and Software Engineering:

A systematic literature review

Muhammad Ovais Ahmad, Jouni Markkula, Markku Oivo, Bolaji Adeyemi
Dept. of information processing science
University of Oulu,
Oulu, Finland
e-mail: {Muhammad.Ahmad, Jouni.Markkula, Markku.Oivo}@ oulu.fi,
Bolaji.Adeyemi@student.oulu.fi

***Abstract*— There is a growing interest about Kanban in software engineering due to its many advantages, such as, reduced lead-time and improved team communication. Kanban originates from Toyota manufacturing and in 2004 it was introduced to software engineering. However, there is lack of a clear explanation of its principles and practices in software engineering. The objective of this study is to explore Kanban in industrial engineering literature using systematic literature review method. The search strategy revealed 1552 papers, of which 52 were identified as primary studies relevant to our research. From the primary studies, five variations of Kanban were identified together with implementation principles and benefits. These were extracted and summarized for the guidance of practitioners interested in adopting Kanban for software development. The findings of this literature review help researchers and practitioners to gain a better understanding of the Kanban and its use in industrial engineering in order to improve its usage in software engineering.**

***Keywords*- Kanban; software development; systematic literature review; lean, agile, Kanban variants.**

I. INTRODUCTION

In the last decade, lean approach in software development is increasingly popular. The aim of the lean approach is to deliver value to customers more effectively and efficiently through the process of finding and eliminating waste, which is a huge impediment to the productivity and quality offered by an organization [39]. Lean approach was first applied in manufacturing industry, devised at Toyota and originally called the Toyota Production System (TPS). According to Ohno [40] TPS was established based on two concepts. The first is "automation with a human touch", which means when a problem occurs, the equipment stops immediately, preventing defective products from being produced. The second concept is "Just-in-Time," which means in each process produces only what is needed by the next process in a continuous flow.

Kanban is a subsystem of TPS, created initially to control inventory levels and the production and supply of components and raw materials [2][14]. Kanban was created to fulfil specific needs of Toyota Company, i.e., to work effectively under specific production and market conditions. Nowadays, Kanban is not only used in manufacturing industries, but also in software development and services, the health sector, and many more domains [2][13]. Kanban facilitates high production volume, high capacity utilization, and reduced production time and work-in-process [14]. Further, Kanban controls the flow of parts along down-stream processing, which creates a "pull" action with material required.

Kanban entered the software development field in 2004, when David Anderson introduced it in practice while assisting a software development team at Microsoft [1][2][3]. Kanban is used to visualise work, limit work in progress, and identify process constraints to achieve flow and yet focus on a single item at a given time [3]. In general, Kanban aims to bring visibility to work, and to enhance communication, collaboration, and integration between software developers, testers, and support teams, resulting in rapid software development and continuous delivery to the customer [1][2][3][4][5][6][7][41]. In software development, the goal of practising Kanban is to visualise and improve the flow of value by optimizing a cycle time, while respecting work in progress limits [1][11][41].

Kanban in software development uses cards to represent work items. Software practitioners have implemented Kanban techniques using physical materials such as sticky notes on a board. Signals are mostly generated from a software work tracking system [1], for example Agile Zen, and Jira. In software development, Kanban has five core principles: *visualise workflow, limit work in progress, measure and manage flow, make process policies explicit, and use models to recognise improvement and opportunities* [1]. Kanban principles are applied using a board which visualises the flow of activities of the process in various columns. Cards are used for each working item on the Kanban board to show its current state. The flow of work items through the process is optimised by limiting the work in progress in each activity column to a maximum number of items that can be pulled into the column. In this manner, the team effectively visualises their workflow, limits work in progress items in each stage, and monitor the cycle time from start to finish.

Kanban is becoming more popular in software development. A strong practitioner-driven movement has emerged supporting its use [3]. Recent studies [2][41] of it in software development shows both benefits and challenges in the adoption of Kanban. The benefits of using Kanban in software development are: a better understanding of the

 ISBN: 978-1-61208-438-1

whole process, improved team communication and coordination with stakeholders, and improved customer satisfaction [2][3][41]. Currently, Kanban is being increasingly adopted to complement Scrum and other agile methods in software processes. With the growing number of studies on Kanban in software development, some have reported a number of challenges, such as hard to manage work in progress, task prioritization, and misunderstanding of core Kanban principles [2][7][8][9]. Notably, evidence of challenges relating to both its implementation and operationalization exist. As Kanban originated in manufacturing industry, by investigate its features in the industrial engineering field literature we can learn how its basic idea is transformed to software engineering.

The purpose of this paper is to systematically review and analyse the Kanban variations, benefits, characteristics, and implementation principles from its industrial engineering context, and to compare them with Kanban in software engineering; in order to find ways to improve its implementation in the software engineering field. To the best of our knowledge, this is the first study that systematically investigates Kanban in industrial engineering and software engineering and how knowledge of Kanban usage in industrial engineering could improve software engineering.

The rest of this paper is organized as follows. Section II describes the research method employed in this study. Section III, describes the results of the review, describes analysis of the results, and provides discussion. Finally, Section IV reports a conclusion and recommendations for further research.

II. RESEARCH METHOD

We designed this systematic literature review by following the guidelines of Kitchenham and Charters [10]. According to these guidelines, we undertook the review in several steps:

- The development of review protocol,
- The identification of inclusion and exclusion criteria,
- A search for relevant studies,
- Data extraction, and
- Synthesis and reporting of results.

The review protocol was developed jointly by the authors of this paper while also carrying out identification and selection of the primary studies on adherence to the specified protocol. All of the steps of the protocol are described below in this section. The objective of the review was to answer the following research questions:

- How does software engineering Kanban differ from industrial engineering Kanban in terms of characteristics?
- How are variants of Kanban and their characteristics and benefits described in industrial engineering?
- How can industrial engineering Kanban implementation principles be used in software engineering?

A. Data Sources and Search Strategies

Four major databases were selected as literature sources: Scopus, IEEE Xplore, Web of Science, and ACM. The rationale for this choice is that these are the relevant databases having large collections of high quality, peer reviewed conferences and journal papers. Relevant studies were searched in these databases by using the following search strings, which were combined with the OR, operator:

1. Pull system AND Kanban
2. Toyota Production System AND Kanban
3. Kanban AND Inventory system
4. Lean AND Kanban
5. Kanban AND (Implementation OR Benefits OR waste elimination)
6. Just-In-Time OR JIT AND Kanban

"Operation OR production" was added at the end of each search string to focus the search to industrial engineering literature.

B. Selection process and inclusion decisions

In Step 1, the titles, abstracts, and keywords of papers were searched using the above mentioned search terms. In Step 1 search resulted in a total of 1,552 papers. In Step 2, duplicate papers were excluded.

In Step 3, two researchers sat together and went through the titles of all studies that resulted from Step 2, to determine their relevance. The systematic review included peer reviewed qualitative, quantitative and simulation research studies published from 1977 up to 2013. Only studies written in English were included. Additionally, editorials, prefaces, correspondence, discussions, lessons learned and expert-opinion papers were also excluded.

In this step, we excluded studies that were clearly not about Kanban in industrial engineering. However, titles are not always clear indicators of what an article is about. In such cases, the articles were included for review in the next step.

In Step 4, each of the remaining papers was assessed with regard to quality and relevance to our study. In assessing the quality of studies, we developed a checklist outlining the major quality criteria expected from the primary studies. We built the list based on quality criteria adapted from Kitchenham and Charters [10]. In this step, two researchers independently reviewed 257 papers. All disagreements were resolved by discussion that included two researchers before proceeding to the next stage.

The evaluation was based on the following criteria: objective of the study, context description, research design, data collection and analysis, and justification of findings. Of the evaluated 257 studies 52 were finally accepted and included as the primary studies for our research. The rest of the papers were excluded because they did not pass the minimum quality threshold.

The following Table I present the systematic review process, which was carried out through this study to identify the primary studies, as well as the number of papers identified at each stage.

 ISBN: 978-1-61208-438-1

TABLE I. STEPS OF THE STUDY SELECTION PROCESS.

Steps	*No. of studies*
Step 1: Search of the bibliographic databases	1552
Step 2: Removing of the duplicates studies	1365
Step 3: Inclusion based on title and abstract	1138
Step 4: Inclusion based on full text scanning	257
Step 5: Quality evaluation based on full paper reading	52

C. Data extraction, synthesis and analysis

Based on the guidance provided in Cruzes and Dyba [12], we extracted three types of data: Kanban description, different variations of Kanban, and benefits of using Kanban. We used a thematic analysis technique. Coding technique was used manually to identify the relevant text in finally included papers while reading the entire paper. All primary studies were categorized by paying close attention to the type of studies which are as follows: conceptual studies, simulation studies, mathematical approaches, surveys, literature reviews, and case studies. Data from all primary studies were extracted by two authors in consensus meetings.

III. RESULTS

We identified 52 peer reviewed primary studies on Kanban to address our research questions. Most of the studies were published in journals (46 or 89%), while six (11%) were published in conferences. The primary studies were categorized on the basis of study type. Out of 52 primary studies 19 (38%) were conceptual studies, 18 (34%) simulation studies, 3 (6%) mathematical approaches, 1 (2%) survey, 7 (13%) reviews, and 4(8%) case studies. Primary studies which are used in this paper are marked with asterisk "*" in the reference list.

A. Kanban in industrial engineering

It appears the first academic paper describing Kanban was published by Japanese researchers in 1977. The title of the paper by Sugimori, Kusunoki, Cho, and Uchikawa [16] is: "Toyota Production System and Kanban System: Materialization of Just-In-Time and Respect-For-Human System." Describing Just in Time (JIT) as a central element of TPS, Sugimori, Kusunoki, Cho, and Uchikawa [16] suggest that JIT is a method whereby production lead times are greatly shortened in order to allow "all processes to produce the necessary parts at the necessary time and have on hand only the minimum stock necessary to hold the processes together". Regarding JIT, Sugimori, Kusunoki, Cho, and Uchikawa [16] consider the following three defining characteristics:

- Levelling of production
- One piece production and conveyance
- Withdrawal by subsequent processes

Further, Sugimori, Kusunoki, Cho, and Uchikawa [16] elaborate that TPS consists of two unique features. The first feature is JIT, which aims to produce only the necessary quantity products at the given time and keeping the inventory (stock at hand) at a minimum level through Kanban. The second feature is to promote respect-for-humans with the ultimate goal of uncovering workers' full potential, through active participation.

Kanban is a Japanese word meaning card or signboard [16], but it can also be a verbal instruction, a light, a flag, or even a hand signal [17][21]. According to Huang and Kusiak [26] Kanban is also known as a `pull' system in the sense that the production of the current stage depends on the demand of the subsequent stages, i.e., the preceding stage must produce only the exact quantity withdrawn by the subsequent manufacturing stage. Typically a Kanban card has information such as part name and part number, quantity designated (the size of the container) in the production process, input areas and output areas [17][18][21].

The key objective of a Kanban system is to deliver the material JIT to the manufacturing workstations, and to pass information to the preceding stage regarding what and how much to produce [26]. According to Sugimori, Kusunoki, Cho, and Uchikawa [16] reasons to use Kanban are: (1) reduction in information processing cost, (2) rapid and precise acquisition of facts, and (3) limiting surplus capacity of preceding shops or stages.

All the primary studies reported that the original Kanban used at Toyota had the following characteristics:

- Kanban system uses two types of signals. First production signals (authorizes a process to produce a fixed amount of product) and second transportation signals (authorizes transporting a fixed amount of product downstream). We use the code "PS" to denote this characteristic.
- Pulled production: The production is pulled based on the inventory level or the scheduling of the last station. We use the code "PP" to denote this characteristic.
- Decentralised control: The control of the production flow is performed through visual control by the employees of each step of the production process. We use the code "DC" to denote this characteristic.
- Limited work in progress: the inventory level is limited in each workstation, which means, buffer capacity depend on the number of signals. We use the code "LW" to denote this characteristic.

To implement original Toyota Kanban there are six principles discussed by Huang and Kusiak [26] and Sugimori, Kusunoki, Cho, and Uchikawa [16] in primary studies, which are as follows:

- Level production (balance the schedule) in order to achieve low variability of the number of parts from one time period to the next [16]. Production levelling can also be referred to reducing the waste. On a production line, as in any process fluctuations in performance can produce waste. When demand is constant production levelling is easy, but when demand fluctuates two approaches can be adopted to handle it: i) demand levelling; and ii) production

 ISBN: 978-1-61208-438-1

levelling through flexible production [26]. To prevent fluctuations in production, it is important to minimize fluctuation in the final assembly line and make production batches as small as possible.

- Avoid complex information and hierarchical control systems on the company floor which creates confusion and disturbs information flow [16]. In such situations, following the Kanban pull system becomes more complex and difficult [26].
- Do not withdraw parts without a Kanban card. Create a strict environment where the Kanban pull system is followed. Do not allow any associate within the production site to withdraw the parts without the Kanban card.
- Withdraw only the parts needed at each stage. In every stage of production withdraw only the parts which are needed for production at the given stage [16][26]. Do not include any additional parts along with the required parts in a production line.
- Do not send defective parts to the succeeding stages. Sending the defective parts to the succeeding stages will increases rework on same parts along with rejection of finished products [16][26].
- Eliminate waste due to over-production, thus produce the exact quantity of parts.

The primary studies reported a number of original Kanban usage benefits, which are as follows:

- Reduced work in progress and Cycle Time: Sugimori, Kusunoki, Cho, and Uchikawa [16] discuss the automotive industry as consisting of multi-stage processes; generally the demand for the items is unpredictable as the process point is further removed from the point of original demand for finished goods. Preceding processes require having excess capacity, and are liable to have waste by over-producing [16]. By limiting releases Kanban regulates work in progress. By Little's Law [15], this also translates into shorter manufacturing cycle times. Kouri, Salmimaa, and Vilpola [37] reported that limiting work in progress helps to consume resources efficiently at a given time. Further, Marek, Elkins, and Smith [28] explained that controlling work in progress helps reduce amount of reworks and financial losses.
- Smoother Production Flow: By dampening fluctuations in work in progress level, a steadier, more predictable output can be achieved [16][26].
- Improved Quality: Working in short queues challenges tolerance of rework because it will quickly shut down the production line [19][20][28]. Short queues reduce the time between creation and detection of a defect. Consequently, Kanban applies pressure for better quality and provides an environment in which to achieve it [30][34][35][38].
- Reduced Cost: Kanban focusses on limiting work in progress levels which eventually helps to reduce total cost [16][19][20][27][38]. Each reduction in work in progress also causes challenges (such as a setup is too long, worker breaks are uncoordinated) in the form of blocking of a line. Solving these problems by lowering the inventory allows production to proceed. This process was widely described via the analogy of lowering the water (inventory) in a river to find the rocks (problems) [16][30][35]. The end result is a more efficient system with lower costs. Kanban reduces inventory cost, inspection cost, unit product cost, and administrative cost [27]. Furthermore, according to Fearon [22] Kanban helps to identify and prioritize problems and opportunities for improvement, and enhances customer and supplier communications.

The following Table II summarizes the five variations of Kanban from primary studies, based on their similarities (in terms of characteristics) mentioned above; and differences (in terms of operationalization) from the original Toyota Kanban. An operational difference will be based on the following points:

1. Inventory variability: During production, quantity of inventory can be varied. In the original Kanban, the inventory level variation is not systematised although some maximum quantities can change between two different planning periods.
2. Physical cards are not used as signals.
3. Modification of the original concept of using two signals, i.e., production and transportation signal.
4. Visual control: Compare to original Kanban use different visual control to gather and apply information related to inventory level and demand.

TABLE II. STEPS OF THE STUDY SELECTION PROCESS.

Variations of Kanban	*Characteristics (similarities)*	*Operational difference*
Generic Kanban	PP, DC, LW	3
Generalised Kanban Control System	PP, DC, LW	3
Extended Kanban Control System	PP, DC, LW	3
Flexible Kanban System	PP, DC,	1, 4
Electronic Kanban	PP, DC, LW	2, 3, 4

Pulled production (PP), Decentralized control (DC), Limited work in progress (LW), Kanban system (PS)

Generic Kanban was proposed by Chang and Yih in 1994 for non-repetitive production environments [23][24][25]. It used generic signals which do not belong to any one part, and thus can be attributed to any item in a workstation. This system requires a waiting time since there is no arbitrator work in progress between workstations. There are signals that if removed do not initiate the production of new parts automatically, instead they wait for a new requirement [23]. Generic Kanban behaves similarly to the push system except that it is more flexible with

 ISBN: 978-1-61208-438-1

respect to system performance and more robust as to the location of the bottleneck [23]. Simulation results showed that the generic Kanban is better than the original Kanban in dynamic environments [23]. The advantage of generic Kanban is that it can be used effectively in environments with unstable demand, as well as in productive environments with variability in processing times [23][24][25].

Generalised Kanban Control System includes the maintenance of buffers to meet the demand instantaneously, and the use of signals to authorize the production and to limit work in progress level [25][33]. The disadvantage of this is the need to define and manage two control parameters per stage, which are the buffer and the number of production order signals [25][33]. The point of difference is that in the Generalised Kanban Control System the transfer of a finished part from a given stage to the next stage and the transfer of demands to the input of this stage may be done independently of one another, whereas in the original Kanban they are done simultaneously [25]. The advantage of Generalised Kanban Control System is that it works effectively when the demand is unstable [25][33].

Extended Kanban Control System is a pulling system proposed for multi-stage manufacturing environments, which works like generalized Kanban with pull production polices, Kanban control system, and base stock control systems [29][32][34]. The difference is that, a work item can only be received in a stage if both production order and free cards are available [29][32][34]. The central idea of this system is that when the demand arrives, it is instantaneously announced to all stations, as in base stock. However, no part is made available without an authorization from the downstream stages [34]. The advantage of Extended Kanban Control System is that it works effectively in an environment with variability processing time [29][32][34].

Flexible Kanban System was introduced to cope with uncertainties and planned/unplanned interruptions [31]. It uses an algorithm to dynamically and systematically manipulate the number of signals in order to offset the blocking and starvation caused by uncertainty (mainly related to demand and processing time) during a production cycle. Gupta, Al-Turki, and Perry [31] summarises Flexible Kanban System as: The idea behind it is to increase the flow of production by judicially manipulating the number of cards. It maintains a minimal number of base level Kanban cards assigned to each station. Extra Kanban cards are added only when needed to improve the system performance and removed as soon as they are no longer needed or when their presence will result in a lowered system performance. That is, we want the extra Kanban cards when the benefits of their presence (e.g., reduced blocking and improved throughput) balance the costs (e.g., increased work in progress and operating costs) [31].

Electronic Kanban is a variation of Kanban with only one modification—the substitution of physical signals by electronic signals [36]. The goal of Electronic Kanban is to introduce an effective means for properly changing the number of cards in the Kanban system [36][37]. One of the most important things in the practical implementation of the system is properly changing the number of cards. Electronic Kanban has many advantages over the original Kanban system in reduced fluctuation and efficient change in the number of cards, faster response to demand change, and effective management of work in progress [36][37]. Electronic Kanban is resulting in improvements in supplier relationships when the systems are used outside the company, by evaluating the supplier's performance instantaneously, and guaranteeing accuracy in acquisition and transmission of amounts [36]. It can be used no matter what the distance between production and operations; it reduces the amount of the company's paperwork, reduces the probability of error associated with signals handling, reduces time to transfer and handle signals, and facilitates new product introduction [36][37].

In the literature, the above variations to Kanban are developed due to competitive industrial environments reflecting unfavourably on the use of the original Kanban system due to the need for greater variety of items, operational standardising difficulties, and demand and processing time instability. Electronic Kanban has more advantages in that it can manage parts ordering and delivery activities more efficiently and effectively than the other variants of Kanban. Additionally, it minimizes operational and logistics issues for a parts supplier or between work stations and complex flow of materials.

B. Kanban learning from industrial engineering to software development

Software development is not a manufacturing activity, because in software development every time we create something new with each development cycle, whereas manufacturing produces the same product over and over again. So, direct mapping of the manufacturing Kanban concept to software development is not logical. In software development, Kanban is used to visualise work, limit work in progress, and identify process constraints to achieve flow and focus on a single item at a given time [3]. The following Table III, compares original Toyota Kanban and software development Kanban characteristics.

TABLE III. COMPARISON OF TOYOTA KANBAN AND SOFTWARE DEVELOPMENT KANBAN CHARACTERISTICS

Kanban Characteristics	***Toyota Kanban [16]***	***Software Development Kanban [1]***
Physical	Yes	Yes
Pull	Yes	Yes
Visual	Yes	Yes
Signal	Yes	Yes
Kaizen	Yes	Yes
Limited work in progress	Yes	Yes
Continuous flow	Yes	Yes
Self-directing	Yes	Yes

Table III, shows that Kanban in software development has all those characteristics which are building blocks of Toyota Kanban and electronic Kanban. Kanban in software development and Toyota Kanban use physical cards to visualise work items and signal the current work in progress. The limited work in progress principle is used to control the work in progress in given time so, as to not exceed the capacity of a system or team. In software development, limiting work in progress, continuous flow, and pull characteristics are not attained by or of themselves. In software development, Kanban focuses more on enabling tasks visual and self-directing; so, as to help the team members become autonomous and improve their own process. To continuously improve the process of continuous flow and to better understand team work in progress limits, daily stand-up meetings are important in communicating information.

In distributed software development teamwork, the electronic Kanban makes it possible to visualise work of remote teams and obtain up-to-date status of projects instantaneously. For software maintenance work, Generic Kanban will work effectively because maintenance teams are dealing with a vast variety of unpredictable and critical tasks, for example, work on isolated or short-time-frame tasks which require quick responses. Generic Kanban has the advantage that it works effectively in environments with unstable demand as well as in productive environments with variability in processing times.

The characteristics of Toyota Kanban in primary studies can be translated in the software development as following:

- Each software development task is represented by a card on Kanban board. Further, these cards signalling the status of activity in the workflow.
- Pulled production: In software development it is refers to trigger the process of producing items only what the customer requested; while restricting to produce the quantity that is required and only when it is needed.
- Decentralised control: In software development, Kanban empower each team member to freely pull the items when their capacity allows. Additionally, everyone is trying to maintain the flow.
- Limited work in progress: In each phase of software development (i.e., development, testing) a limit on work in progress is applied which shows the capacity and signal when it is full or ready to pull upcoming items.

In software development, we can implement the original Toyota Kanban discussed by Huang and Kusiak [26] and Sugimori, Kusunoki, Cho and Uchikawa [16] as follows:

- Kanban is a visual management tool that shares a mental model; visualise the work, the workflow, and the business risks to the whole team and or organisation [1]. According to Al-Baik and Miller [41] Kanban helps in enhancing visual control that facilitated and supported the decision-making process.
- To implement Kanban in software development is to start with what you do now [1][41]. Avoid complex information and hierarchical structure in assigning the tasks. Use the Kanban board and allow the development team to pull the tasks automatically. Further, too much controlling of task assignment should be avoided because it creates confusion, disturb information flow and makes difficult to use pull technique [1][11][41].
- Carefully apply work in progress limits on each stage of software development (i.e., development, testing) [1][41]. Limited work in progress is the means by which we can create a pull system which balances capacity and demand through the value stream [16]. Further, it implies that pull system is implemented to the workflow. In software development, this means that upstream work can be made available, but it is the team member's responsibility to decide when to take it. The act of pulling the work is a signal for more upstream work to be processed.
- A card needs to be assigned to every customer request and should be visualised on a Kanban board [1][41]. Do not withdraw tasks without a Kanban card. Create a strict environment where the Kanban pull system is followed. Do not allow any team member within the development team to withdraw the tasks without the Kanban card.
- Pull out only the tasks that are of high priority [1][41].
- Do not send partially done tasks. By sending to succeeding stages will increases rework on the same task along with rejection of finished products [1][11][41].
- Eliminate waste due to over-producing by working on the exact tasks which are needed or requested by the customer [1][41].

IV. CONCLUSION

Kanban is a subsystem of the Toyota Production System (TPS), which was created to control inventory levels, production, and supply of components. Kanban entered software development in 2004, when David Anderson introduced it in practice while assisting a software development team at Microsoft.

To learn from industrial engineering examples of Kanban usage, systematic review and analysis of Kanban variations, benefits, and implementation principles was conducted. Searches of the literature identified 1552 studies of which 53 were found to be studies of acceptable credibility and relevance. Most of the primary studies (72%) are conceptual or use simulation techniques to investigate Kanban in industrial engineering.

 ISBN: 978-1-61208-438-1

This study reported the original Toyota Kanban and its five variations from industrial engineering literature. Toyota Kanban which is also adopted in software engineering has the four basic characteristics: pulled production, decentralized control, limited work in progress, and two types of signals (i.e., production signals and transportation signals). The Toyota Kanban and its five variations have similar characteristics fundamentally concerned with signals use and manipulation in terms of number or quantity. For example, electronic Kanban has one modification—the substitution of physical signals by electronic signals.

In software development work, the Toyota Kanban concept is adopted. For distributed software development, electronic Kanban is more suitable. Whereas, based on the Generic Kanban characteristics, it is judged to be more effective for work which has unstable demands and variability in processing times (i.e., software maintenance and support). In maintenance work, customers' requests come on a daily basis, and tasks are constantly prioritised based on severity. The advantage of Generic Kanban is that it works effectively when the demand is unstable. In software development it is used to achieve better process control (keeping continuous flow while limiting work in progress) and better process improvement (makes the flow visible and stimulates Kaizen). Kanban in both industrial engineering and software engineering yields benefits such as smoother production or development flow, reduced cycle time, and improved quality.

Future studies are needed to explore Kanban variation limitations, disadvantages, and challenges in their usage. Further, it is recommended to conduct detailed comparative studies on Kanban variations along with the Kanban used in software development.

ACKNOWLEDGMENT

This research was carried out within the DIGILE Need for Speed program, and partially funded by Tekes (the Finnish Funding Agency for Technology and Innovation) and a Nokia foundation scholarship.

REFERENCES

[1] D. J. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, 2010.

[2] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review". Proceedings of the 39th Euromicro Conference Series on Software Engineering and Advanced Applications (SEAA 2013), pp. 9-16.

[3] M. O. Ahmad, J. Markkula, and M. Oivo, and P. Kuvaja, "Usage of Kanban in Software Companies - An empirical study on motivation, benefits and challenges". Proceeding of the 9th International Conference on Software Engineering Advances (2014). pp, 150-155.

[4] G. Concas, M. I. Lunesu, M. Marchesi and H. Zhang, "Simulation of software maintenance process, with and without a work-in-process limit," Journal of Software: Evolution and Process (2013), vol. 25, pp. 1225-1248.

[5] D. I. Sjøberg, A. Johnsen and J. Solberg, "Quantifying the effect of using Kanban versus Scrum: A case study," Software, IEEE, (2012), vol. 29, pp. 47-53.

[6] J. Prochazka, "Agile support and maintenance of IT services," in Information Systems Development Anonymous Springer, (2011), pp. 597-609.

[7] X. Wang, K. Conboy and O. Cawley, ""Leagile" software development: An experience report analysis of the application of lean approaches in agile software development," J. Syst. Software, (2012) vol.85, pp.1287-1299.

[8] E. Corona, and F.E. Pani, "A review of lean-kanban approaches in the software development". WSEAS Transactions on Information Science and Applications, (2013), vol. 10(1), pp. 1-13.

[9] N. Nikitina, and M. Kajko-Mattsson,"Developer-driven big-bang process transition from scrum to kanban". Proceedings of the International Conference on Software and Systems Process, (2011), pp. 159-168.

[10] B. Kitchenham, and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering" EBSE Technical Report, EBSE-2007-01.

[11] M. Burrows, Kanban from the inside. Blue Hole Press, Sequim, Washington. 2014.

[12] D. S. Cruzes, and T. Dyba, Research synthesis in software engineering: A tertiary study', Information and Software Technology, (2011) vol. 53, no. 5, pp. 440–455.

[13] Bennet Vallet (2014). Kanban at Scale – A Siemens Success Story, InfoQ. [Online] Available from: http://www.infoq.com/articles/kanban-siemens-health-services 2015.07.03

[14] C.S. Kumar, and R. Panneerselvam, Literature review of JIT-KANBAN system. The International Journal of Advanced Manufacturing Technology, (2007), 32(3-4), pp. 393-408.

[15] J. D. Little, and S.C. Graves, Little's law. In Building Intuition Springer US. (2008), pp. 81-100.

[16] Y. Sugimori, K. Kusunoki, F. Cho, and S. Uchikawa, "Toyota production system and kanban system materialization of just-in-time and respect-for-human system", The International Journal of Production Research, (1977), vol. 15(6), pp. 553-564. *

[17] O. Kimura, and H. Terada, "Design and analysis of pull system, a method of multi-stage production control", The International Journal of Production Research, (1981), vol. 19(3), pp. 241-253. *

[18] G.R. Bitran, and L. Chang, "A mathematical programming approach to a deterministic kanban system". Management Science, (1987), vol. 33(4), pp. 427-441. *

[19] H. Dyck, R.A. Johnson, and J. Varzandeh, "Transforming a traditional manufacturing system into a just-in-time system with kanban". Proceedings of the 20th Conference on Winter Simulation, (1988), pp. 616-623. *

[20] M. Gravel, and W.L Price, "Using the kanban in a job shop environment". The International Journal of Production Research, (1988), vol. 26(6), pp. 1105-1118. *

[21] Jr. Esparrago and R. A, "Kanban. Production and Inventory" Management Journal, (1988), vol. 29(1), pp. 6-10. *

[22] P.A. Fearon, "Inventory controlled environment (I.C.E.) just-in-time at national semiconductor". Proceedings of the Advanced Semiconductor Manufacturing Conference and Workshop, (1993), pp. 34-38. *

[23] T. Chang and Y. Yih, "Generic kanban systems for dynamic environments". The International Journal of Production Research, (1994), vol. 32(4), pp. 889-902. *

[24] T. Chang and Y. Yih, "Determining the number of kanbans and lotsizes in a generic kanban system: A simulated annealing approach". The International Journal of Production Research, (1994), vol. 32(8). *

[25] Y. Frein, M. Di Mascolo, and Y. Dallery, "On the design of generalized kanban control systems". International Journal of Operations and Production Management, (1995), vol. 15(9), pp. 158-184. *

[26] C. Huang, and A. Kusiak, "Overview of kanban systems". International journal of computer integrated manufacturing, (1996), col. 9(3), pp. 169-189. *

[27] L. Sriparavastu and T. Gupta, "An empirical study of just-in-time and total quality management principles implementation in manufacturing firms in the USA". International Journal of Operations and Production Management, (1997), vol. 17(12), pp. 1215-1232. *

[28] R.P. Marek, D.A. Elkins, and D.R. Smith, "Manufacturing controls: Understanding the fundamentals of kanban and CONWIP pull systems using simulation". Proceedings of the 33nd Conference on Winter Simulation, (2001), pp. 921-929. *

[29] J. Bollon, M. Di Mascolo, and Y. Frein, "Unified framework for describing and comparing the dynamics of pull control policies". Annals of Operations Research, (2004), vol. 125(1-4), pp. 21-45. *

 ISBN: 978-1-61208-438-1

[30] E. Kizilkaya, and S.M. Gupta, "Material flow control and scheduling in a disassembly environment". Computers and Industrial Engineering, (1998), vol. 35(1), pp. 93-96. *
[31] S.M. Gupta, Y.A. Al-Turki, and R.F. Perry, "Flexible kanban system". International Journal of Operations and Production Management, (1999), vol. 19(10), pp. 1065-1093. *
[32] G. Liberopoulos, and Y. Dallery, "A unified framework for pull control mechanisms in multi-stage manufacturing systems". Annals of Operations Research, (2000), vol. 93(1-4), pp. 325-355. *
[33] C. Duri, Y. Frein, and M. Di Mascolo, "Comparison among three pull control policies: Kanban, base stock, and generalized kanban". Annals of Operations Research, (2000), vol. 93(1-4), pp. 41-69. *
[34] Y. Dallery, and G. Liberopoulos, "Extended kanban control system: Combining kanban and base stock. IIE Transactions, (2000), vol. 32(4), pp. 369-386. *
[35] K. Takahashi, and N. Nakamura, "Decentralized reactive kanban system". European Journal of Operational Research, (2002), vo1. 39(2), pp. 262-276. *
[36] S. Kotani, "Optimal method for changing the number of kanbans in the e-kanban system and its applications". International Journal of Production Research, (2007), vol. 45(24), pp. 5789-5809. *
[37] I. Kouri, T. Salmimaa, and I. Vilpola, "The principles and planning process of an electronic kanban system". Novel algorithms and techniques in telecommunications, automation and industrial electronics, (2008), pp. 99-104. *
[38] M. Akturk, and F. Erhun, "An overview of design and operational issues of kanban systems". International Journal of Production Research, (1999), vol. 37(17), pp. 3859-3881. *
[39] J.K. Liker, M. Hoseus, "Toyota culture: the heart and soul of the Toyota Way". (2008), McGraw-Hill, New York, USA
[40] O, Taiichi. "Toyota production system: beyond large-scale production". (1988), Productivity press.
[41] O. Al-Baik and J. Miller, "The kanban approach, between agility and leanness: a systematic review" Empirical Software Engineering. (2014), pp. 1-37.

 ISBN: 978-1-61208-438-1

Efficient ETL+Q for Automatic Scalability in Big or Small Data Scenarios

Pedro Martins, Maryam Abbasi, Pedro Furtado
University of Coimbra
Department of Informatics
Coimbra, Portugal
email: {pmom, maryam, pnf}@dei.uc.pt

***Abstract*—In this paper, we investigate the problem of providing scalability to data Extraction, Transformation, Load and Querying (ETL+Q) process of data warehouses. In general, data loading, transformation and integration are heavy tasks that are performed only periodically. Parallel architectures and mechanisms are able to optimize the ETL process by speeding-up each part of the pipeline process as more performance is needed. We propose an approach to enable the automatic scalability and freshness of any data warehouse and ETL+Q process, suitable for smallData and bigData business. A general framework for testing and implementing the system was developed to provide solutions for each part of the ETL+Q automatic scalability. The results show that the proposed system is capable of handling scalability to provide the desired processing speed for both near-real-time results and offline ETL+Q processing.**

***Keywords*-Algorithms; architecture; Scalability; ETL; freshness; high-rate; performance; scale; parallel processing.**

I. INTRODUCTION

ETL tools are special purpose software used to populate a data warehouse with up-to-date, clean records from one or more sources. The majority of current ETL tools organize such operations as a workflow. At the logical level, the E (Extract) can be considered as a capture of data-flow from the sources with more than one high-rate throughput. T (Transform) represents transforming and cleansing data in order to be distributed and replicated across many processes and ultimately, L(Load) convey by loading the data into data warehouses to be stored and queried. For implementing these type of systems besides knowing all of these steps, the acknowledge of user regarding the scalability issues is essential, which the ETL+Q (queries) might be introduced.

When defining the ETL+Q the user must consider the existence of data sources, where and how the data is extracted to be transformed, loading into the data warehouse and finally the data warehouse schema; each of these steps requires different processing capacity, resources and data treatment. Moreover, the ETL is never so linear and it is more complex than it seems. Most often the data volume is too large and one single extraction node is not sufficient. Thus, more nodes must be added to extract the data and extraction policies from the sources such as round-robin or on-demand are necessary.

After extraction, data must be re-directed and distributed across the available transformation nodes, again since transformation involves heavy duty tasks (heavier than extraction), more than one node should be present to assure acceptable execution/transformation times. Consequently, once more new data distribution policies must be added. After the data transformed and ready to be load, the load period time and a load time control must be scheduled. Which means that the data have to be held between the transformation and loading process in some buffer. Eventually, regarding the data warehouse schema, the entire data will not fit into a single node, and if it fits, it will not be possible to execute queries within acceptable time ranges. Thus, more than one data warehouse node is necessary with a specific schema which allows to distribute, replicate, and query the data within an acceptable time frame.

In this paper, we study how to provide parallel ETL+Q scalability with ingress high-data-rate in big data and small data warehouses. We propose a set of mechanisms and algorithms, to parallelize and scale each part of the entire ETL+Q process, which later will be included in an auto-scale (in and out) ETL+Q framework. The presented results prove that the proposed mechanisms are able to scale when necessary.

Section II approaches the related work in the field. Section III describes the proposed architecture. Section IV describes the experimental setup and obtained results. Finally, Section V concludes the presented work and introduces some future research lines.

II. RELATED WORK

Works in the area of ETL scheduling includes efforts towards the optimization of the entire ETL workflows [6] and of individual operators in terms of algebraic optimization, e.g., joins or data sort operations. However, many works focus on complex optimization details that only apply to very specific cases. Munoz et al. [3] focus on finding approaches for the automatic code generation of ETL processes which is aligning the modeling of ETL processes in data warehouse with Model Driven Architecture (MDA) by formally defining a set of Query, View, Transformation (QVT) transformations. ETLMR [2] is an academic tool which builds the ETL processes on top of Map-Reduce to parallelize the ETL operation on commodity computers. ETLMR does not have its own data storage (note that the

 ISBN: 978-1-61208-438-1

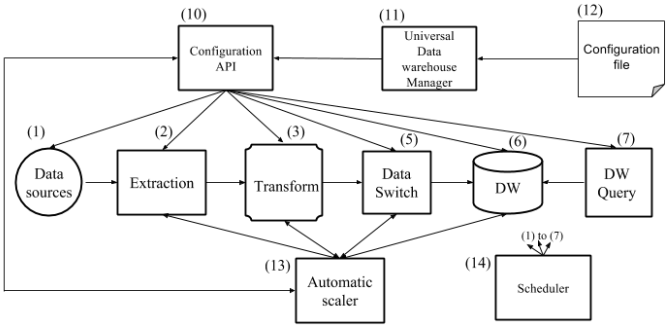


Figure 1. Automatic ETL+Q scalability

offline dimension store is only for speedup purpose), but is an ETL tool suitable for processing large scale data in parallel. ETLMR provides a configuration file to declare dimensions, facts, User Defined Functions (UDFs), and other run-time parameters. ETLMR toll has the same problem as the MapReduce architectures, too much hardware resources are required to guaranty basic performance.

In [5], the authors consider the problem of data flow partitioning for achieving real-time ETL. The approach makes choices based on a variety of trade-offs, such as freshness, recoverability and fault-tolerance, by considering various techniques. In this approach, partitioning can be based on round-robin (RR), hash (HS), range (RG), random, modulus, copy, and others [7].

In [1] the authors describe Liquid, a data integration stack that provides low latency data access to support near real-time in addition to batch applications.

There is a vast related work in ETL field. Although main related problems studied in the past include the scheduling of concurrent updates and queries in real-time warehousing and the scheduling of operators in data streams management systems. However, we argue that a fresher look is needed in the context of ETL technology. The issue is no longer the scalability cost/price, but rather the complexity it adds to the system. Previews presented recent works in the field do not address in detail how to scale each part of the ETL+Q and do not regard the automatic scalability to make ETL scalability easy and automatic. We focus on offering scalability for each part of the ETL pipeline process, without the nightmare of operators relocation and complex execution plans. Our main focus is automatic scalability to provide the users desired performance with minimum complexity and implementations. In addition, we also support queries execution.

III. Architecture

In this section, we describe the main components of the proposed architecture for ETL+Q scalability. Figure 1 shows the main components to achieve automatic scalability.

- All components from (1) to (7) are part of the Extract, Transform, Load and query (ETL+Q) process. All can auto scale automatically when more performance is necessary.
- The ”Automatic Scaler” (13), is the node responsible for performance monitoring and scaling the system when is necessary.
- The ”Configuration file” (12) represents the location where all user configurations are defined by the user.
- The ”Universal Data Warehouse Manager” (11), based on the configurations provided by the user and using the available ”Configurations API” (10), sets the system to perform according with the desired parameters and algorithms. The ”Universal Data Warehouse Manager” (11), also sets the configuration parameters for automatic scalability at (13) and the policies to be applied by the ”Scheduler” (14).
- The automatic scaler module (13), based on time bounds configurations and limit amounts of resources to use (mainly memory) scales the ETL pipeline modules.
- The ”Configuration API” (10), is an access interface which allows to configure each part of the proposed Universal Data Warehouse architecture, automatically or manually by the user.
- Finally, the ”Scheduler” (14), is responsible for applying the data transfer policies between components (e.g., control the on-demand data transfers).

All these components when set to interact together are able to provide automatic scalability to the ETL and to the data warehouses processes without the need for the user to concern about its scalability or management.

IV. Experimental setup and results

In this section, we describe the experimental setup, and experimental results to show that the proposed system, AScale, is able to scale and load balance data in small and big data scenarios for near real-time and offline ETL+Q.

The experimental tests were performed using 30 computers, denominated as nodes, with the following characteristics: Processor Intel Core i5-5300U Processor (3M Cache, up to 3.40 GHz); Memory 16GB DDR3; Disk: western digital 1TB 7500rpm; Ethernet connection 1Gbit/sec; Connection switch: SMC SMCOST16, 16 Ethernet ports, 1Gbit/sec; Windows 7 enterprise edition 64 bits; Java JDK 8; Netbeans 8.0.2; Oracle Database 11g Release 1 for Microsoft Windows (X64) - used in each data warehouse nodes; PostgreSQL 9.4 - used for look ups during the transformation process; TPC-H benchmark - representing the operational log data used at the extraction nodes. This is possible since TPC-H data is still normalized; SSB (star schema benchmark) benchmark - representing the data warehouse. The SSB is the star-schema representation of TPC-H data. Data transformations consist loading from the data sources the ”lineitem” and ”order” TPC-H data logs and besides the transformation applied to achieve the SSB benchmark

 ISBN: 978-1-61208-438-1

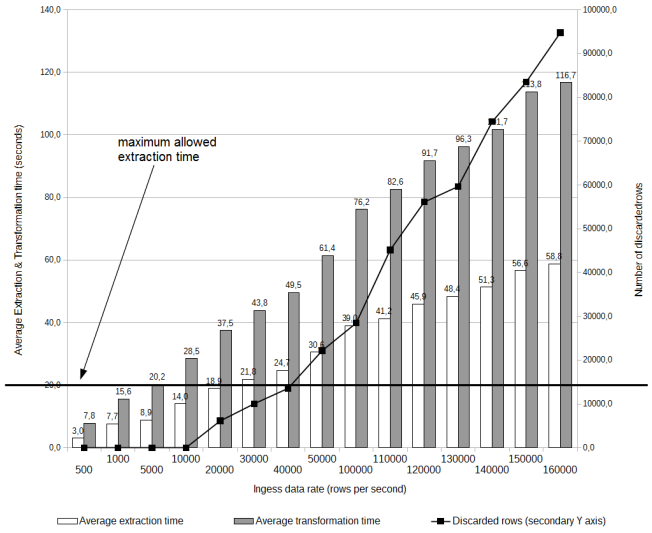


Figure 2. Extract and transform without automatic scalability

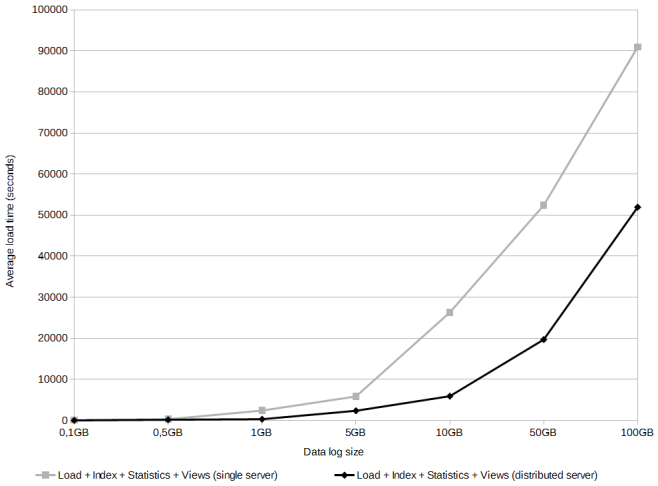


Figure 3. Loading data, one server vs two servers

star schema [4] we added some data quality verification and cleansing.

A. *Performance limitations without automatic scalability*

In this Section, we test both ETL and data warehouse scalability needs when the entire ETL process is deployed without automatic scalability options. The system is stressed with increasing data-rates until it is unable to handle the ETL and query processing in reasonable time. Automatic scalability which we evaluate in following sections, is designed to handle this problem.

The following deployment is considered: One machine to extract, transform data and store the data warehouse; extraction frequency is set to perform every 30 seconds; desired maximum allowed extraction time, 20 seconds; data load is performed in offline periods.

Based on this scenario, we show the limit situation in which performance degrades significantly, justifying the need to scale the ETL (i.e., parts of it) or data warehouse.

Extraction & transformation: Considering only extraction and transformation, using a single node, Figure 2 shows: in the left Y axis is represented the average extraction and transformation time in seconds; in the right Y axis is represented the number of discarded rows (data that was not extracted and not transformed); in the X axis we show the data-rate in rows per-second; white bars represent extraction time; gray bars represent the transformation time; lines represent the average number of discarded rows (corresponding values in the right axes). For this experiment, we generated log data (data to be extracted) at a rate λ per second. Increasing values of λ were tested and the results are shown in Figure 2.

Extraction is performed every 30 seconds. This means that in 30 seconds there is 30x more data to extract. Extraction must be done in 20 seconds maximum. As the data-rate increases, a single node is unable to handle so much data. At a data-rate of 20.000 rows per second, buffer queues become full and data starts being discarded at sources because the extraction time is too slow. The transformation process is slower than transformation, requiring more resources to perform at the same speed as the extraction.

Loading the data warehouse: Figure 3 shows the load time as the size of the logs is increased. It also compares the time taken with single single node versus two nodes. All times were obtained with the following load method: destroy all indexes and views, load data, create indexes, update statistics and update views; data was distributed by replicating and partition the tables. Differences are noticeable when loading more than 10GB. When adding two data warehouse nodes, performance improves and the load time becomes almost less than half.

- The Y axis represents the average load time in seconds and the X axis represents the loaded data size in GB.
- The black line represents two servers and the grey line represents one server.

Query execution: Figure 4 shows the average query execution time for a set of tested workload (using the SSB benchmark queries): workload 1, 10 sessions, 5 Queries (Q1.1, Q1.2, Q2.1, Q3.1, Q4.1); workload 2, 50 sessions, 5 Queries (Q1.1, Q1.2, Q2.1, Q3.1, Q4.1); workload 3, 10 sessions, 13 Queries (All); workload 4, 50 sessions, 13 Queries (All); for all workloads, queries were executed in a random order; the desired maximum query execution time was set to 60 seconds.

The Y axis shows the average execution time in seconds. The X axis shows the data size in GB. Each bar represents the average execution time per query fro each workload. Note that, Y axis scale is logarithmic for better results representation.

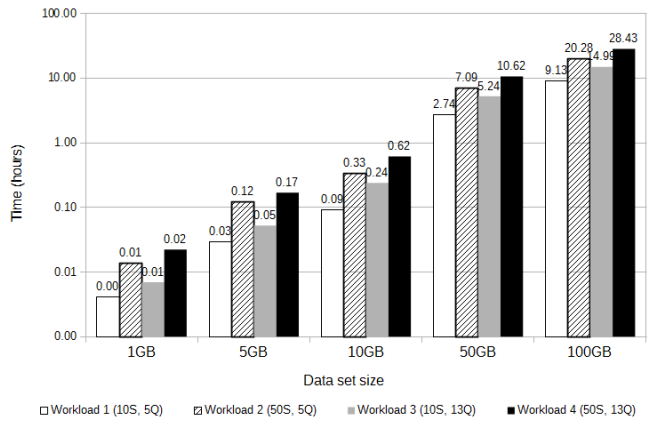


Figure 4. Average query time for different data sizes and number of sessions

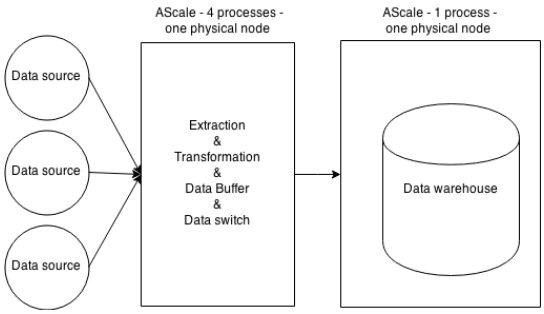


Figure 5. AScale for simple scenarios

Depending on the data size, number of queries and number of simultaneous sessions (e.g., number of simultaneous users), execution time can vary from a few seconds to a very significant number of hours or days, especially when considering large data sizes and simultaneous sessions or both. In these results, and referring to 10GB and 50GB, we see that an increase of 5x of the data size resulted in an increase of approximately 20x in response time. An increase in the number of 5x resulted in an increase of approximately 2x in query response time.

B. Typical data warehouse scenarios

In this section we evaluate AScale in a scenario where because of log sizes and limited resources, data load takes too long to perform without scaling.

We start with only two nodes (two physical machines), one for handling extraction and transformation, the other to hold the data warehouse as shown in Figure 5. AScale is setup to monitor the system and scale when needed.

Data is extracted from sources, transformed and loaded only during a predefined period (e.g., night), to be available for analysis the next day. The maximum extraction, transformation and load time, all together cannot take longer than 9 hours (e.g., from 0am until 9am). AScale was

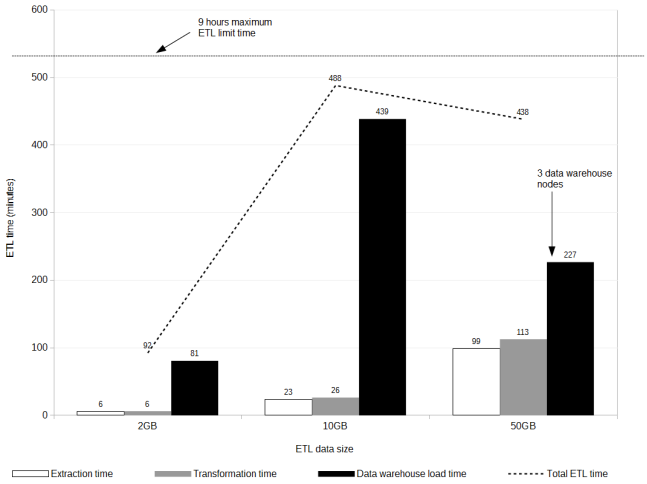


Figure 6. AScale, 9 hours limit ETL time

configured with an extraction frequency of every 24 hours and a maximum duration of 4 hours, a transformation queue with a limit size of 10GB and data warehouse loads were configured for every 24 hours, with a maximum duration of 9 hours. Note that, the entire ETL process was set for a maximum duration of 9 hours.

The experimental results form Figure 6 show the total AScale ETL time using two nodes (two physical machines), one for extract, transform, data buffer and data switch, other for the data warehouse. Up to 10GB the ETL process can be handled within the desired time windows. However, when increasing to 50GB, 9 hours are no longer enough to perform the full ETL process. In this situation, the data warehouse load process (load, update indexes, update views) using one node (average load time 873 minutes) and two nodes (average load time 483 minutes) exceeds the desired time window. When scaled to 3 nodes, by adding one data warehouse node, the ETL process returns to the desired time bound.

The extraction and transformation process were never scaled, since they were able to perform within the desired time, the same for the data buffer and data switch that were able to handle all data within defined bounds.

C. ETL offline scalability with huge data sizes

In this section we create an experimental setup to stress AScale under extreme data rate conditions. The objective is to test scaling each part of the pipeline.

For this experiment we did the following configured: E (extraction) was set to perform every 1 hour with 30 minutes maximum extraction time, T (transformation) queue maximum size was configured to 500MB, and L (load) frequency to every 24 hours with a maximum duration of 5 hours.

 ISBN: 978-1-61208-438-1

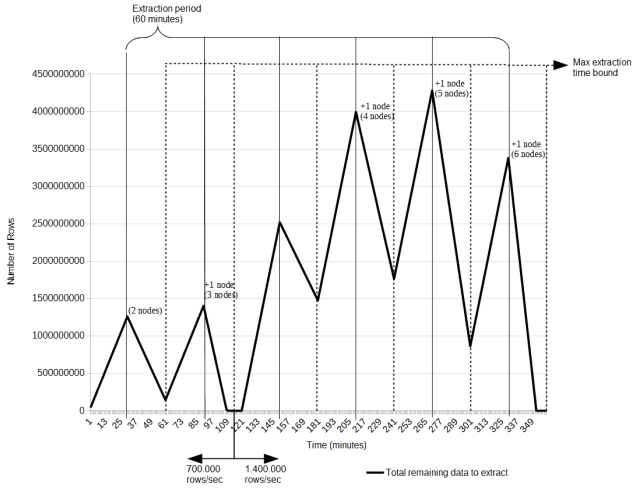


Figure 7. Extraction (60 minutes frequency and 30 minutes maximum extraction time)

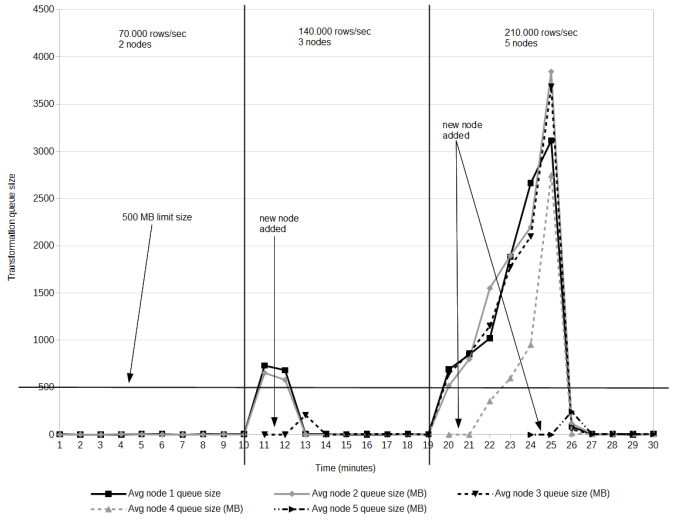


Figure 8. Offline, transformation scale-out

Extraction: Figure 7 shows the AScale extraction process when using an extraction frequency of 60 seconds and 30 seconds for the maximum extraction time.

In Figure 7, we show the followings: the left Y axis the number of rows, the X axis is represented the time in minutes and the black line represents the total number or rows left to be extracted at each extraction period. By analyzing the results from Figure 7 we conclude that the extraction process is able to scale efficiently when more computational power is necessary. However, if the data rate increases very fast in a small time window AScale requires additional extraction cycles to restore the normal extraction frequency.

Transformation: In Figure 8 are shown the transformation scale-out tests based on nodes ingress data queue size.

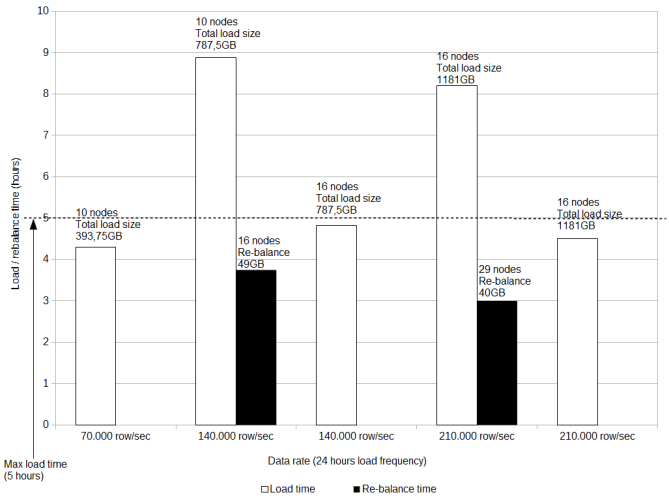


Figure 9. Offline, load scale-out

Every time a queue fills-up until the maximum configured size AScale automatically scale-out. This monitoring process allows to scale-out very fast, even if the data rate increases suddenly. Each scale out took only an average of 2 minutes, referring to the copy and replication of the staging area. Experimental results show that AScale transformation can be scaled-out more than one node in a very short time frame.

Load: AScale load process is done at the end of each load cycle that did not respected the maximum load time. The number of nodes to add is calculated linearly based on previews load time. For instance, if load time using 10 nodes was 9 hours. To load in 5 hours we need x nodes, estimated in equation 1.

$$\frac{loadTime}{targetTime} \times n \quad (1)$$

Where "loadTime" represents the last load time, "targetTime", represents the desired load time and "n" represents the current number of nodes.

Figure 9 shows the data warehouse nodes scalability time and data (re)balance time. We conclude that the data warehouse nodes can be scaled efficiently in a relatively short period of time given the large amounts of data being considered.

D. Near-Real-time DW scalability and Freshness

In this section we assess the scale-out and scale-in abilities of the proposed framework in near-real-time scenarios requiring data to be always updated and available to be queried (i.e., data freshness).

The near-real-time scenario was set-up with: E (extraction) and L (load) were set to perform every 2 seconds; T (transformation) was configured with a maximum queue size of 500MB; the load process was made in batches of

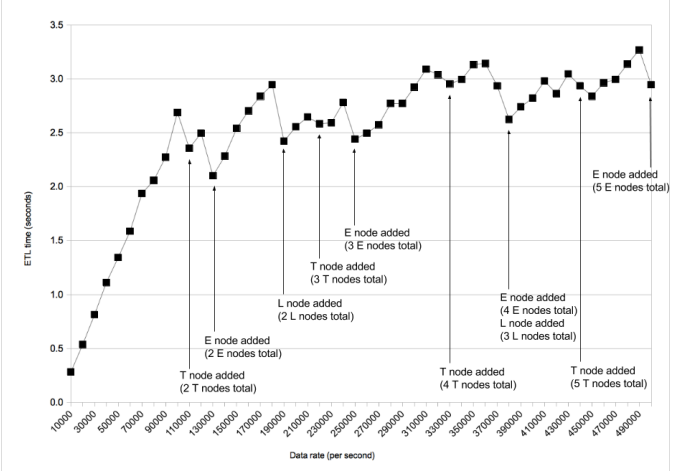


Figure 10. Near-real-time, full ETL system scale-out

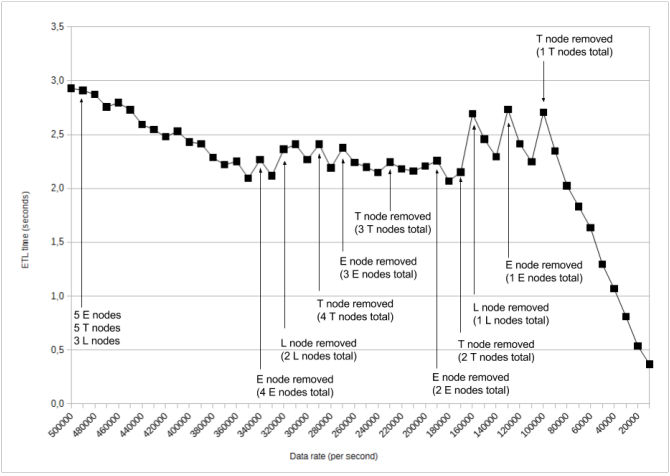


Figure 11. Near-real-time, full ETL system scale-in

100MB maximum size. The ETL process is allowed to take 3 seconds.

Figures 10 and 11 show AScale, scaling-out and scaling-in automatically, respectively, to deliver the configured near-real-time ETL time bounds, while the data rate increases/decreases. The system objective was set to deliver the ETL process in 3 seconds. The charts show the scale-out and scale-in of each part of the AScale, obtained by adding and removing nodes when necessary. A total of 7 data sources were used/removed gradually, each one delivering a maximum average of 70.000 rows/sec. AScale used a total of 12 nodes to deliver the configured time bounds.

Near-real-time scale-out results in Figure 10 show that, as the data-rate increases and parts of the ETL pipeline become overloaded, by using all proposed monitoring mechanisms in each part of the AScale framework, each individual module scales to offer more performance where and when necessary.

Near-real-time scale-in results in Figure 11 show the instants when the current number of nodes is no longer necessary to ensure the desired performance, leading to some nodes removal (i.e., set as ready nodes in stand-by, to be used in other parts).

V. Conclusions & Future work

In this work we proposed mechanisms to achieve automatic scalability for complex ETL+Q, offering the possibility to the users to think solely in the conceptual ETL+Q models and implementations for a single server.

Tests demonstrate that the proposed techniques are able to scale-out and scale-in when necessary to assure the necessary efficiency. Future work includes real-time event processing integration oriented to alarm and fraud detection. Other future work included making an visual drag and drop interface, improve monitoring and scale decision algorithms, and finally provide usability comparisons with other academic tools. A beta version of the framework is being prepared for public release.

References

[1] N. Ferreira, P. Martins, and P. Furtado. Near real-time with traditional data warehouse architectures: factors and how-to. In *Proceedings of the 17th International Database Engineering & Applications Symposium*, pages 68–75. ACM, 2013.

[2] X. Liu. *Data warehousing technologies for large-scale and right-time data*. PhD thesis, dissertation, Faculty of Engineering and Science at Aalborg University, Denmark, 2012.

[3] L. Muñoz, J.-N. Mazón, and J. Trujillo. Automatic generation of etl processes from conceptual models. In *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, pages 33–40. ACM, 2009.

[4] P. E. O'Neil, E. J. O'Neil, and X. Chen. The star schema benchmark (ssb). *Pat*, 2007.

[5] A. Simitsis, C. Gupta, S. Wang, and U. Dayal. Partitioning real-time etl workflows, 2010.

[6] A. Simitsis, P. Vassiliadis, and T. Sellis. Optimizing etl processes in data warehouses. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 564–575. IEEE, 2005.

[7] P. Vassiliadis and A. Simitsis. Near real time etl. In *New Trends in Data Warehousing and Data Analysis*, pages 1–31. Springer, 2009.

 ISBN: 978-1-61208-438-1

The Role of People and Sensors in the Development of Smart Cities: A Systematic Literature Review

Italberto Figueira Dantas
UFPI - Federal University of Piaui
Teresina, Brazil
Email: italberto@ufpi.edu.br

Felipe Silva Ferraz
CESAR - Recife Center for Advanced Studies and Systems
Recife, Brazil
Email: fsf@cesar.org.br

***Abstract*—Over the last few years, with rapid population growth in the biggest cities of the world, issues like air pollution, water scarcity, and intense traffic conditions, have become more evident. Trying to mitigate them, the concept of Smart City was presented, which uses technology and human resources to manage urban resources in a sustainable manner. As new scientific researches about this phenomenon are being carried out, the importance of both human and technological resources remain implicit in the development of Smart Cities. But, it is not clear what role is occupied in the different stages of evolution, which leads a city to be considered Smart. Given this context, in this paper we used the SLR method to analyze scientific publications, and thus to determine what is the role of the human factor, represented by people, and technological, represented by sensors, in the development of smart cities. We also created the overview of the scientific research, and can identify the most and least studied areas among those addressed in this study.**

Keywords–Smart City; Sensors; IoT; People; Open Innovation.

I. INTRODUCTION

In the last century, the number of cities in the world with more than 1 million inhabitants jumped from 20 to 450. In 2007, the number of people living in urban centers exceeded 50% of the world population and it is estimated that this number will reach 70% by the year 2050 [1]. The fast growth of the population in urban centers makes us face problems like the deterioration of public transportation services, decrease of air quality and increase of unemployment, etc. [2]. When dealing with these issues, it is necessary to use creativity, human resources, and cooperation between the different areas of the society and good ideas [3].

Technology is a way to solve the problems caused by the population growth in urban centers. Some, such as the IoT (Internet of Things), are used in the context of Smart cities due to their potential to improve the life quality of the population [4]. Not only investments in technology should be taken into account when thinking about sustaining Smart Cities, but human and social capital must be used as a fuel for economic growth and high quality of life by using the natural resources in an intelligent way, through government policies that involve the society [3].

In the current scenario, every initiative is important if it helps understand the growth of great urban centers and how it occurs, as well as what measures should be adopted to extinguish the problems caused by it, from a scientific and economic point of view. Through a Systematic Literature Review (SLR), a panorama of the current scientific research will be presented, dealing with the relationship between people and sensors with the development of modern cities, which are Smart Cities.

The problem considered in this work may be described the following way: There is a large number of works approaching the participation of people and the use of sensors in Smart Cities. There is a need to visualize the current panorama of this research line in a broad way in order to identify open issues, as well as identifying researches to be used for accelerating the studies of this field.

In this work, we intend to provide the current panorama of the scientific researches made in the field of Smart Cities that talk about their development based on the participation of people and the use of sensors, based on the Systematic Review methodology, according to what Kitchenham [5] proposed.

In this work, the term ”development” is used to identify the initiatives of creation of new Smart Cities, as well as the improvement of those that already possess management of natural resources and monitoring of basic services and wish to improve them. Also, the term ”sensors” includes sensors that receive information on the environment, actors that execute actions and transform the environment they live in and other technologies related.

This work is organized as follows: Section II presents the necessary concepts to understand the rest of the research; Section III presents the methodology used; Section IV presents the research protocol used; Section V exposes the results obtained from the execution phase of the protocol and presents the analysis of the results and finally, Section VI exposes the conclusions, suggestions for future researches and the final considerations.

II. SMART CITY

The term Smart City has been used for 20 years and it has evolved due to concerns about the service supply and resource consumption [6], which are increasing with the growth of urban centers. According to Nam et al. [2], the term Smart City can be approached according to three perspectives: (1) Technological Elements: Hardware and software infrastructure; (2) Human Elements: creativity, diversity and education; (3) Institutional Elements: governance and politics. A city may be considered as a smart one when investments in social and human capital and Information Technologies infrastructure transform environmental growth and improve quality of life through participative governance [3].

The use of the word smart as a label for future cities is not by chance. In marketing words, smartness focuses on the

 ISBN: 978-1-61208-438-1

perspective of the user [7], and it is related to a fast mind, with efficient answers. Smart Cities need to adapt themselves rapidly to the needs of their citizens and provide customized interfaces [2]. Technology is a way to reach it. In this context, Smart Cities can be defined, according to Kehua et al. [8], as the use of information and technologies of communication to measure, analyze and integrate the information of the main services of a city. This way, Smart Cities may respond intelligently to different types of need, including daily maintenance, environmental protection, public safety and commercial activities.

A. IoT

At the same time the access to the Internet becomes easier, computer devices are getting smaller and more popular through mobile devices with the evolution of the industry over the years. IoT can be defined as a global network infrastructure with the ability to self-set itself based on interoperable communication patterns and protocols, through physical and virtual things have an identity, physical attributes and smart virtual interfaces integrated through an information network [9].

Smart Things networks promise to revolutionize the monitoring of environments in a great variety of application domains due to their reliability, efficiency, flexibility, low cost and easy installation [10].

B. Smart City and People

The concept of Smart Cities only makes sense when we talk about the presence of people as target audience of the benefits achieved through the use of technology and other ways to improve quality of life. According to Washburn et al. [11], what makes a city smart is the use of technology to provide basic services for the citizens in an efficient way.

Smart Cities are human cities with multiple opportunities to explore the human potential and turn life into a more productive life [2]. Technological issue, very present in the current conception of smart cities, becomes secondary in the views centered in the social and human part.

III. Methodology

In this section, we present the methodology used.

SLR is, almost always, the initial phase of any research [12]. The purpose is to accumulate more knowledge about the subject the research is about. SLR goes beyond a simple literary review because it uses scientific methodology and provides a way to integrate studies, creating generalizations about the subject.

This modality of study is a way to identify, evaluate and interpret a relevant part of the research about a specific matter of the research, area or phenomena of interest [5].

According to Petersen et al. [13], the need to carry out systematic reviews is that as a specific area matures, a lot of research is made, which generates a great number of primary works that need to be summarized.

Systematic review uses the review protocol as its main tool, which defines a series of steps that must be followed in order to get to a conclusion [12]. These steps must be very well defined so other people can reproduce the same process to validate the study.

The systematic review model used in this work, is based on the proposal of Kitchenham [5] and performed by Oliveira et al. [14], Budgen et al. [15] and Ribeiro et al. [16]. Biolchini et al. [12] proposed a model used on the protocol.

The model is based on three main phases: (1) Planning: the research objectives and the research question are defined; (2) Execution: primary studies are identified, selected and evaluated according to the criteria of inclusion and exclusion defined during the Planning phase; (3) Presentation of Results: presentation of the report with the information obtained during the previous phase.

Described phases may seem sequential, however, many tasks performed in each one of the phases can be started in the Planning phase and concluded only during the Execution phase.

Biolchini et al. [12] describe the three phases through a chart, as seen in Figure 1.

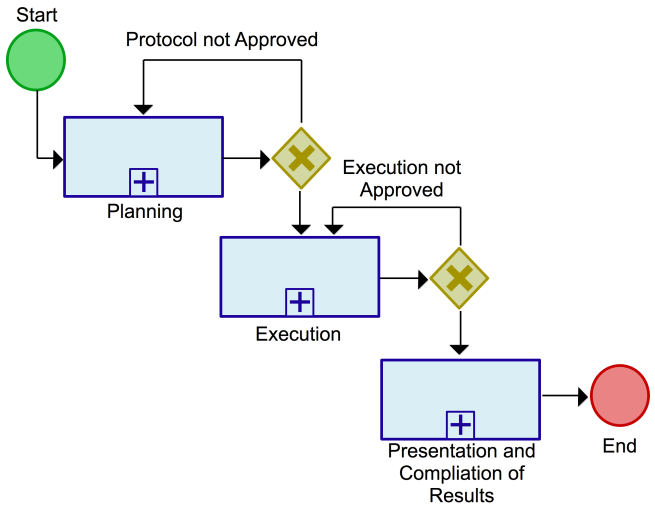


Figure 1. Three stages SLR process.

According to them [12], the process of systematic review can be divided into five steps:

- Problem formulation. Aims to define what kind of evidences will be included in the review. On this step, the criteria that will help the researcher define which studies are relevant for his research should be set;
- Data collection. The objective is to define which will be the sources to acquire evidence;
- Data evaluation. Aims to define which information will be used in the research. Quality criteria must be applied in order to distinguish the valid studies;
- Analysis and interpretation. Synthesizing data is the main goal to create generalization about the researched subject in order to determine if it can be solved or not;
- Conclusion and presentation of the results. The objective is to decide which information will be presented in the final report, since not all the information obtained in the data analysis are relevant for the research.

 ISBN: 978-1-61208-438-1

IV. Protocol Definition

Protocol definition is the first step of the research development. Using the problem related in the Introduction of this work, a research question was elaborated with the objective of conducing the research procedures. We may define this question as: **Q01** (Question 01) What is the role of people and sensors in the development of Smart Cities?

Novais et al. [17] suggests the division of the research question into sub-questions in order to provide greater coverage. Thus, Q01 was subdivided: **Q1.1**. What is the influence of the participation of people or use of sensors in the study?; **Q1.2**. What is the level of previous infrastructure so the study proposal is viable?; **Q1.3**. What stage of social organization the city needs to be in so the study is viable?; **Q1.4**. What is the contribution of the study for the scientific community?; **Q1.5**. How was the study validated?; **Q.1.6**. What was the main feature of the city affected by the study?.

After defining what will be observed, the next step is selecting the research bases of the research, which were:

- Compendex (CPE);
- IEEExplore (EXP);
- Science Direct (SDI);
- Scopus (SCO).

Besides the chosen research bases, a manual research through the references of the publications was made, based on the initial researches, restricted to articles, thesis and dissertations in English.

A research expression was defined to be used in the consultations in each one of the research bases, using the research question as a guide. The expression used was: **EX01** - (((digital OR smart) AND (city)) AND (development OR creation OR expansion)) AND (iot OR ict) AND (people OR citizen)).

After researching in the chosen bases, the results obtained were exported and stored.

The approach to select the primary studies follows the guidelines of Barcellos et al. [18], which suggest only the need to define exclusion criterion in order to select relevant studies. This way, four steps were defined and for two of them, exclusion criterion were defined as well. Details are presented in Table I. The stages can be described as follows:

- **ST01**. The search must be done in the selected bases SCO, EXP, SDI and CPE using the search expression EX01. Results obtained must be stored to facilitate future consultation;
- **ST02**. This stage eliminates primary studies that do not attend the purposes of this study;
- **ST03**. Publications not excluded on the previous step will be re-evaluated based on their title, abstract or full text;
- **ST04**. Search for referenced of the remaining publications must be made.

Data extraction consists in taking relevant information that may help solving the research question through the reading of the text and the metadata.

A test before the final protocol execution must be conducted to assure it is correct, in order to identify possible failures introduced during their definition. The test must be carried out in a reduced portion of the total bases to be used in the conduction of the research [18].

TABLE I. Exclusion Criteria.

Step	Criteria Code	Description
ST02	EC01	Repeated publications.
	EC02	Publications of the same author presenting similar content.
	EC03	Studies which text cannot be obtained.
ST03	EC04	The title indicates the study deals with a different subject from this work.
	EC05	The abstract indicates the study deals with a different subject from this work.
	EC06	The text indicates the study deals with a different subject from this work.

A protocol test was conducted using the CPE and SDI bases with this purpose. Results obtained in the execution of the test were left out; however, they were satisfactory and enabled the protocol execution to be conducted without any changes.

A. Quality Assessment

According to Kitchenham [5], quality criterion can be used to help analyzing and resuming the data obtained, identifying subgroups among the selected studies. This way, a checklist was set with some items that must be applied to the selected studies after executing the ST03 stage.

Quality assessment will help determine if the quality of the analyzed studies influences on the results of the publications. The assessment can be made in parallel to the activity of reading publications.

Issues related to evaluating the quality of the studies are presented in Table II.

TABLE II. Quality Criteria.

ID	Question	Answer
QC01	Are the objectives clear?	Yes/No/NA
QC02	Was there a validation of the proposal?	Yes/No/NA
QC03	Was data collection correctly made?	Yes/No/NA
QC04	Is the purpose of the analysis clear?	Yes/No/NA
QC05	Were the questions asked in the study answered?	Yes/No/NA
QC06	Were the results reported negative?	Yes/No/NA
QC07	Does the study explicitly explore the research question?	Yes/No/NA

To ensure that the selection of publications is performed correctly, each of which must be evaluated by more than one researcher.

V. Results

After the protocol approval, the other stages were executed. The result can be seen in Table III. In this table we can notice that, in the first stage ST01 the number of publications is rather high, a total of 1121 publications, which were evaluated in the following steps according to the selection criteria. In the stage ST02, 61 publications were excluded and , in the stage ST03, the number of excluded publications was 1030.

After applying the exclusion criterion, the manual search for references of the selected publications was made and the general results can be seen in Table IV, as well as the manual search. This table presents a summary of the selection of publications result. The publications included in the stage ST01 were found, so as the publications included

 ISBN: 978-1-61208-438-1

in the stage ST04, matching with the manual search. The exclusions realized in the stages ST02 and ST03 are also detailed, classified according to the search tool that have been used.

45 publications were obtained after executing all the steps. Table V shows all the obtained publications. In this table, each publication is identified by a code. The code of the search tool used to find the publication, the stage in which the publications are and the year of the publication is also showed.

TABLE III. PROTOCOL EXECUTION RESULTS OF THE FIRST THREE STAGES.

Step	Criteria	SCO	EXP	SDI	CPE
ST01		349	648	78	46
		EXC	EXC	EXC	EXC
ST02	CR01	1	38	1	2
	CR02	0	0	0	0
	CR03	4	12	2	1
ST03	CR04	297	531	69	38
	CR05	22	46	5	3
	CR06	5	14	0	0
Total Exclusion		329	641	77	44
Total Remaining		20	7	1	2

TABLE IV. TOTAL PUBLICATIONS BY DATABASE AND MANUAL SEARCH.

	SCO	EXP	SDI	CPE	MSC	Total
Included	349	648	78	46	15	1136
Excluded	329	641	77	44	0	1091
Total	20	7	1	2	15	45

Results of the evaluation of the publications in each one of the sub-questions were compared with the other results. Therefore, deep analysis of the obtained data could be made, once the isolated observation of the results is not enough for the research question to be cleared.

Crossings between the research sub-questions are numbered in Table VI.

A. Question Q1.1 Analysis

During these two phases of development of the Smart Cities, the Sensor element stands out in the Chaotic and Managed levels of organization.

In the initial level of organization, the People element is the one with more highlight. This happens due to the approach of human resource in many publications as a collaborator in data production, along with the sensors.

Last two levels of technological organization, Integrated and Optimized, were the less approached levels among the analyzed publications. A small amount of cities around the world with characteristics that put them in these two last levels is what partly explains this fact.

The first level of social organization, the Chaotic, was the one that obtained the highest number of publications, which is explained by crowdsensing and other data collection techniques. In the last two levels, the People element obtained a higher number of publications.

The result of this research points out that when a society is not organized to sustain initiatives to create a Smart City, Technology is the most expressive element to start the process,

TABLE V. SELECTED PUBLICATIONS.

Code	Engine	Step	Pub. Year	Pub.
PB10	CPE	ST03	2013	[19]
PB13	CPE	ST03	2014	[20]
PB52	SDI	ST03	2014	[21]
PB136	SCO	ST03	2015	[22]
PB144	SCO	ST03	2014	[23]
PB159	SCO	ST03	2014	[24]
PB165	SCO	ST03	2014	[25]
PB170	SCO	ST03	2014	[26]
PB171	SCO	ST03	2014	[27]
PB192	SCO	ST03	2014	[28]
PB195	SCO	ST03	2014	[29]
PB210	SCO	ST03	2014	[30]
PB323	SCO	ST03	2013	[31]
PB334	SCO	ST03	2013	[32]
PB341	SCO	ST03	2013	[33]
PB375	SCO	ST03	2012	[34]
PB389	SCO	ST03	2012	[35]
PB410	SCO	ST03	2012	[36]
PB430	SCO	ST03	2011	[37]
PB436	SCO	ST03	2011	[38]
PB438	SCO	ST03	2011	[39]
PB453	SCO	ST03	2010	[40]
PB467	SCO	ST03	2010	[41]
PB504	EXP	ST03	2013	[42]
PB518	EXP	ST03	2010	[43]
PB638	EXP	ST03	2012	[44]
PB708	EXP	ST03	2014	[45]
PB1024	EXP	ST03	2010	[46]
PB1042	EXP	ST03	2013	[47]
PB1050	EXP	ST04	2011	[48]
PB1127	MSC	ST04	2011	[49]
PB1134	MSC	ST04	2010	[50]
PB1135	MSC	ST04	2011	[2]
PB1139	MSC	ST04	2011	[51]
PB1141	MSC	ST04	2011	[52]
PB1146	MSC	ST04	2011	[53]
PB1148	MSC	ST04	2012	[54]
PB1149	MSC	ST04	2011	[55]
PB1150	MSC	ST04	2013	[56]
PB1151	MSC	ST04	2011	[57]
PB1152	MSC	ST04	2013	[58]
PB1157	MSC	ST04	2011	[59]
PB1158	MSC	ST04	2013	[60]
PB1159	MSC	ST04	2013	[61]
PB1160	MSC	ST04	2011	[62]

TABLE VI. COMPARISON BETWEEN RESEARCH QUESTIONS.

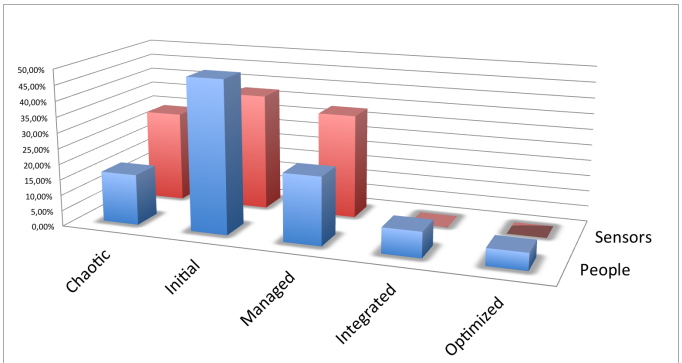
Sub-Question	Compared Sub-Question
Q1.1	1.2, Q1.3, Q1.4, Q1.5, Q1.6
Q1.4	Q1.5
Q1.6	Q1.2, Q1.3, Q1.4

generating more engagement of the population in the subsequent levels, when there is already a favorable environment.

Thus, the result obtained through the analysis of the charts in Figures 2 and 3 indicate that in cities with no infrastructure and control of their resources nor promote participation of the population in the decision-making processes,the administration of the city prefer to invest their efforts to acquire technologies that allow us to go forward towards a Smart City. Then, technology is an enabler of future innovation in the field of the city.

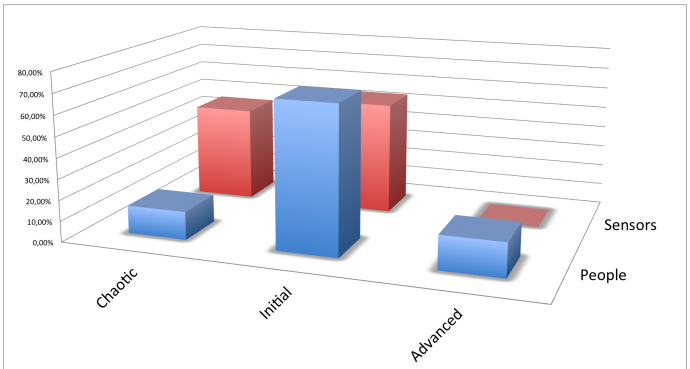
When there are available infrastructure and the engagement of the population in the city affairs, even if it is in a precarious way, it is possible to notice that human resource is the most discussed issue, indicating the influence it may have on the development of the city.

Figure 4 presents the chart generated by the results of the



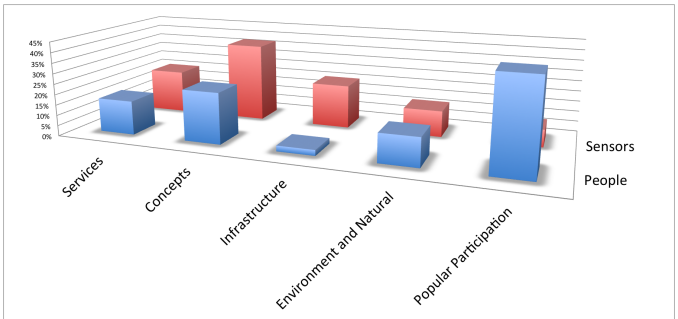
	Chaotic	Initial	Managed	Integrated	Optimized
People	16,21%	48,64%	21,62%	8,10%	5,40%
Sensors	29,16%	37,50%	33,33%	0%	0%

Figure 2. Result of questions Q1.1 and Q1.2.



	Chaotic	Initial	Advanced
People	13,51%	70,27%	16,22%
Sensors	45,83%	54,17%	0,00%

Figure 3. Result of questions Q1.1 and Q1.3.



	Services	Concepts	Infrastructure	Environment and Natural Resources	Popular Participation
People	16%	24%	3%	14%	43%
Sensors	21%	38%	21%	13%	8%

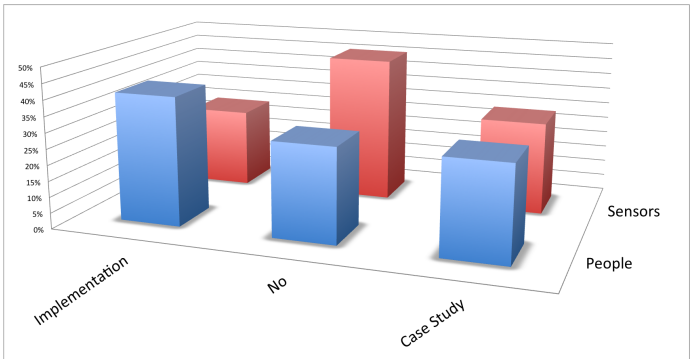
Figure 4. Result of questions Q1.1 and Q1.4.

sub-questions Q1.1 and Q1.4. We can see the elements People and Sensors in it, distributed according to the scientific contributions made. Generally speaking, the technological issue was the most influent in all categories, except the last two, Environment and Natural Resources and Popular Participation.

The result presented reinforces the statement of Chourabi et al. [63] that the study of people and communities is critical in the context of Smart Cities, however, it is being neglected.

The element People presented a small advantage in the category Environment and Human Resources, demonstrating the human perception of the environment is being explored in the monitoring of natural resources and urban environment, being as important as the sensors in the information acquisition. The same element was the most influential in the category Popular Participation, an awaited result due to its direct relationship between people and concepts, such as crowdsensing, open innovation, crowdsourcing and open participation.

Figure 5 represents the chart from crossing the results of the sub-questions Q1.1 and Q1.5. We can see the elements People and Sensors in it, distributed according to the types of validation of each publication.



	Implementation	No	Case Study
People	41%	30%	30%
Sensors	25%	46%	29%

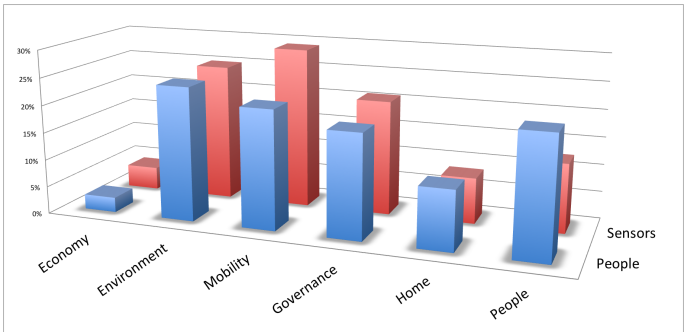
Figure 5. Result of questions Q1.1 and Q1.5.

The element People was the one with greater highlight in the publications that adopted Implementation and Case Study as a way of validation, obtaining only 1% (one percentage point) more than Sensors in the Case Study way. This lower number of the element Sensors related to human perspective is because many publications about the technological perspective maintained themselves in more theoretical subjects, with no need of validation that would fit one of the two validation categories defined in this work.

Figure 6 presents the chart from crossing the results of the sub-questions Q1.1 and Q1.6. We can see the elements People and Sensors distributed according to the areas of the Smart Cities of the analyzed publications.

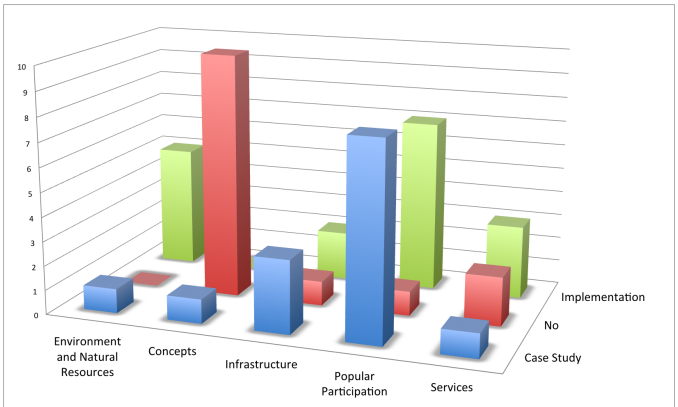
Data presented followed a trend. In the first four areas (Economy, Environment, Mobility, Governance), sensors were the most influent, while in the areas of Housing and People it was the element People the most influent. Generally speaking, results of the analysis of the combination of these two sub-questions may indicate there should have more balance when approaching both elements in every area. The least influenced

 ISBN: 978-1-61208-438-1



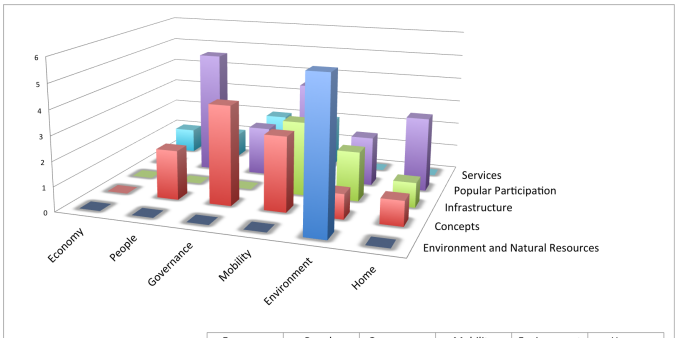
	Economy	Environment	Mobility	Governance	Home	People
People	3%	24%	22%	19%	11%	22%
Sensors	4%	25%	29%	21%	8%	13%

Figure 6. Result of questions Q1.1 and Q1.6.



	Environment and Natural Resources	Concepts	Infrastructure	Popular Participation	Services
Case Study	1	1	3	8	1
No	0	10	1	1	2
Implementation	5	0	2	7	3

Figure 7. Result of questions Q1.4 and Q1.5.



	Economy	People	Governance	Mobility	Environment	Home
Environment and Natural Resources	0	0	0	0	6	0
Concepts	0	2	4	3	1	1
Infrastructure	0	0	0	3	2	1
Popular Participation	0	5	2	4	2	3
Services	1	1	2	2	0	0

Figure 8. Result of questions Q1.4 and Q1.6.

area was Economy, while Mobility was the most. It can be explained by the low event of publications about economical matters, while there were many talking about technological issues.

B. *Question Q1.4 Analysis*

Analyzing the chart of Figure 7 that presents the results of the sub-questions Q1.4 and Q1.5 together, it is expected to be able to determine which are the most used ways of validation in each research area. Firstly, the greatest part of the non-validated publications talked about the theme of this research in a theoretical way or those about works in progress. Popular Participation was the area with more validated publications through implementation and case studies.

In a general way, Implementation was the most used way to validate the efforts, which made it a little ahead the Case Study. It may indicate that Implementation is the most used way of validation because there are not many options to test the previously available proposals, which made the researchers responsible for developing their own tools to validate their proposals. Testbeds and Living Labs are initiatives that may help.

Analyzing the chart of Figure 8 presenting the results of the sub-questions Q1.4 and Q1.6 together, the intention is to identify the most researched areas of the Smart Cities and those that need more attention, taking into account the defined research areas.

Initially, it is possible to notice the concepts of the economic elements were not explored. The research area of Infrastructure did not emphasize technological elements of the areas of Governance, People and Economy and there were also no researches about services in the areas of Housing and Environment. Those numbers suggest more research should be developed in these areas.

Second, the result presents coherent numbers between the two analyzed dimensions, like the cases of the areas of Environment and Natural Resources and the area of Environment, between the areas of Infrastructure and Mobility and between the areas of Popular Participation and People.

As for the rest of the results, it is possible to observe that concepts about all areas of Smart Cities were researched and Popular Participation was also in their agendas. Economy was the only exception in both cases, when it was not mentioned.

In a general way, it is possible to conclude that more researches about the economic features of Smart Cities are necessary.

C. *Question Q1.6 Analysis*

Due to the cross-analysis of the result of sub-question Q1.6 with sub-questions Q1.3 and Q1.3, it is expected to determine which areas of the Smart Cities are more researched during the phases of technological and social evolution.

Result of crossing sub-questions Q1.2 and Q1.6 is shown in the chart of Figure 9. Governance was the area that received more attention, followed by People, Mobility and environment, as observed in the chart. Housing and Economy were not mentioned. This result may point to a city in the Chaotic level as Governance as the area that should get more attention, since it deals with the decision making about the future of the city.

Only Governance was not mentioned in an initial level, while Environment was the most discussed area followed by

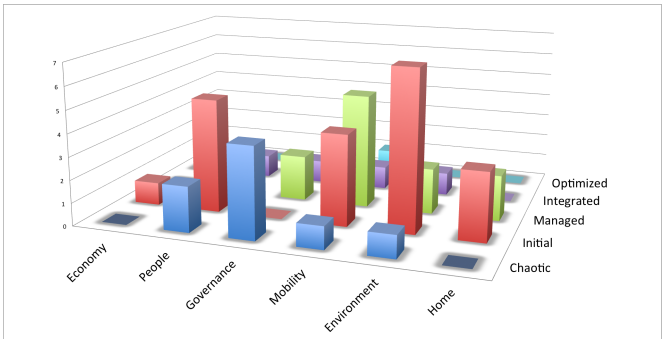
People, Mobility, Housing and Economy. This result may indicate that, in an initial level, Environment is the level that should get more attention.

Mobility was the area that received more attention in a Managed level, indicating that when there is infrastructure, the trend is to try improving it.

Integrated and Managed levels were the least explored among all the evaluated publications. For this reason, it is not possible to conclude anything from the presented chart.

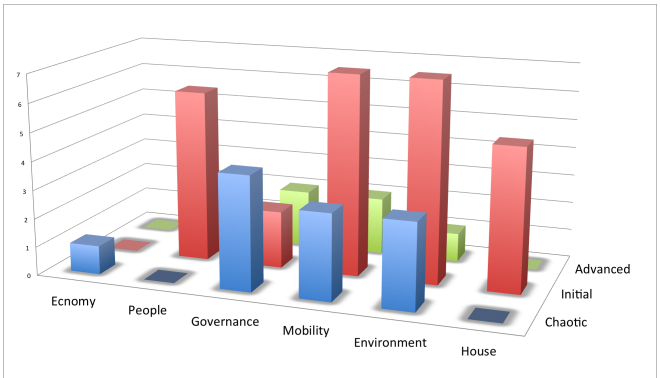
Figure 10 shows a chart that presents the result of the crossing between sub-questions Q1.3 and Q1.6.

Through the chart is possible to notice that in the Chaotic organization level there is more concern with the areas of Governance, Mobility and Environment. However, the number of studies about the People area in this level was zero. According to its definition, in this stage the level of social organization is zero, which demonstrates the evaluated studies talk about an environment where the Popular Participation is already in a more elevated level.



	Economy	People	Governance	Mobility	Environment	Home
Chaotic	0	2	4	1	1	0
Initial	1	5	0	4	7	3
Managed	0	0	2	5	2	2
Integrated	0	1	1	1	1	0
Optimized	0	0	1	1	0	0

Figure 9. Result of questions Q1.2 and Q1.6.



	Ecnomy	People	Governance	Mobility	Environment	House
Chaotic	1	0	4	3	3	0
Initial	0	6	2	7	7	5
Advanced	0	2	2	2	1	0

Figure 10. Result of questions Q1.3 and Q1.6.

The initial level concentrates a bigger number of publications. In this level, Economy was the only area not explored due to the low incidence of publications about this area. Most explored areas in this level were Mobility and Environment, indicating that caring for the environment and technology used in the city are considered more important in this level.

D. Quality Analysis

As defined in the revision protocol, a qualitative evaluation of the 45 selected publications was made in order to avoid biases in the selected publications. Results of this analysis are shown in Table VI.

TABLE VII. RESUME OF QUALITY ANALYSIS RESULTS.

Criteria	Yes	Not	Not Applicable
CQ01	100%	0%	0%
CQ02	71,11%	0%	28,80%
CQ03	71,11%	0%	28,88%
CQ04	71,11%	0%	28,88%
CQ05	100%	0%	0%
CQ06	2,22%	97,77%	0%
CQ07	24,44%	97,77%	0%

Through the results analyzed it was possible to notice that the quality of the selected publications was not good, which did not influence negatively on the analysis of the publications.

VI. Conclusion and Future Work

SLR was the method used in this work, to create a panorama of the scientific research about the participation of people and use of sensors to develop Smart Cities. Based on the work of Kitchenham [5], a revision protocol was developed so the SLR could be executed.

Results obtained during the protocol test show it could be used in a consistent way to reach the objectives of this study. Through the research sub-questions, defined during the elaboration of the revision protocol, it was possible to direct the analysis of the publications to reach the general and specific objectives defined for this study.

The general objective was to provide the current panorama of the scientific research made in the area of the Smart Cities that explore their development based on the participation of people and use of sensors. We reached it by making a bi-dimensional analysis of the research sub-questions results. We could notice that people and sensors have great importance when developing Smart Cities. Their importance changes according to the level of technological and social development each city is in, as well as the needs each city defines as priority.

It was also possible to classify the studies based on different technical and scientific criteria, as well as identifying many areas that need more attention of the scientific area when it comes to popular participation and use of sensors in the development of Smart Cities. Results obtained can show a direction for future researches in most needed areas.

The economic aspect was showed pretty deficient, from almost all analyzed points of view. So, this theme needs to be deeply analyzed, in order to understand better the relationship with the cities technological and social elements.

This research was realized considering just a period of five years of study. A new protocol execution can be made, with a bigger range, in order to analyze the evolution of

 ISBN: 978-1-61208-438-1

the researched aspects with the development of new scientific studies about Smart Cities, making it possible to enhance the knowledge about how people and sensors influence the Smart Cities development, drawing an evolutionary profile.

REFERENCES

[1] C. E. A. Mulligan and M. Olsson, "Architectural Implications of Smart City Business Models : An Evolutionary Perspective," IEEE Communications Magazine, vol. 51, no. 6, Jun. 2013, pp. 80–85.

[2] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," in Proceedings of the 12th Annual International Digital Government Research Conference on Digital Government Innovation in Challenging Times - dg.o '11. New York, New York, USA: ACM Press, 2011, pp. 282–291.

[3] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in Europe," Journal of Urban Technology, vol. 18, no. 2, Apr. 2011, pp. 65–82.

[4] T. Yashiro, S. Kobayashi, N. Koshizuka, and K. Sakamura, "An Internet of Things (IoT) architecture for embedded appliances," in 2013 IEEE Region 10 Humanitarian Technology Conference. IEEE, Aug. 2013, pp. 314–319.

[5] B. A. Kitchenham, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Keele University, Keele, Tech. Rep., 2007.

[6] A. Bartoli, J. Hernández-Serrano, M. Soriano, M. Dohler, and A. Kountouris, "Security and Privacy in your Smart City," in Proceedings of Barcelona Smart Cities Congress 2011, 2011, pp. 1–6.

[7] C. Klein and G. Kaefer, "From Smart Homes to Smart Cities: Opportunities and Challenges from an Industrial Perspective," in Next Generation Teletraffic and Wired/Wireless Advanced Networking, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5174, pp. 260–260.

[8] K. Su, J. Li, and H. Fu, "Smart city and the applications," in 2011 International Conference on Electronics, Communications and Control (ICECC). IEEE, Sep. 2011, pp. 1028–1031.

[9] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, Vision and challenges for realising the Internet of Things, 1st ed., H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, Eds. Luxemburgo: Publications Office of the European Union, 2010, no. 1.

[10] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 6, no. 2, 2002, pp. 28–36.

[11] D. Washburn et al., "Helping CIOs Understand Smart City Initiatives," Cambridge University, Tech. Rep., 2010.

[12] J. Biolchini, P. G. Mian, A. Candida, and C. Natali, "Systematic Review in Software Engineering," Universidade Federal do Rio de Janeiro, Rio de Janeiro, Tech. Rep. May, 2005.

[13] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," in 12th International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 71–80.

[14] L. B. R. Oliveira, F. S. Osório, and E. Y. Nakagawa, "A Systematic Review on Service-Oriented Robotic Systems Development," ICMC/Univ. of So Paulo, Tech. Rep., 2012.

[15] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, "Using Mapping Studies in Software Engineering," PPIG'08: 20th Annual Meeting of the Psichology of Programming Interest Group, vol. 2, 2007, pp. 195–204.

[16] F. Ribeiro, F. S. Ferraz, M. Carolina, G. Henrique, and S. Alexandre, "Big Data Solutions For Urban Environments A Systematic Review," ALLDATA 2015, The First International Conference on Big Data, Small Data, Linked Data and Open Data, 2015, pp. 22–28.

[17] R. L. Novais, A. Torres, T. S. Mendes, M. Mendonça, and N. Zazworka, "Software evolution visualization: A systematic mapping study," Information and Software Technology, vol. 55, no. 11, Nov. 2013, pp. 1860–1883.

[18] M. P. Barcellos, R. Falbo, and R. Rocha, "A Strategy for Measurement of Software and Evaluation of Bases of Measures for Statistical Control of Processes of Software at High Maturity Organizations," Ph.D Thesis, Federal Univ. of Rio de Janeiro, 2009.

[19] D. López-de Ipiña, S. Vanhecke, O. Peña, T. De Nies, and E. Mannens, "Citizen-Centric Linked Data Apps for Smart Cities," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013, vol. 8276 LNCS, pp. 70–77.

[20] L. Roo et al., "Mobile Crowdsourcing Older People s Opinions to Enhance Liveability in Regional City Centres," no. April, 2014, pp. 21–24.

[21] L. Atzori, D. Carboni, and A. Iera, "Smart things in the social loop: Paradigms, technologies, and potentials," Ad Hoc Networks, vol. 18, 2014, pp. 121–132.

[22] C. Klonner, C. Barron, P. Neis, and B. Höfle, "Updating digital elevation models via change detection and fusion of human and remote sensor data in urban environments," International Journal of Digital Earth, vol. 8, no. 2, Feb. 2015, pp. 153–171.

[23] D. S. Gallo, C. Cardonha, P. Avegliano, and T. C. Carvalho, "Taxonomy of Citizen Sensing for Intelligent Urban Infrastructures," IEEE Sensors Journal, vol. 14, no. 12, Dec. 2014, pp. 4154–4164.

[24] A. Cenedese, A. Zanella, L. Vangelista, and M. Zorzi, "Padova Smart City: An urban Internet of Things experimentation," in Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014. IEEE, Jun. 2014, pp. 1–6.

[25] S. Fang, L. D. Xu, Y. Zhu, J. Ahati, H. Pei, J. Yan, and Z. Liu, "An Integrated System for Regional Environmental Monitoring and Management Based on Internet of Things," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, May 2014, pp. 1596–1605.

[26] E. Theodoridis, G. Mylonas, V. Gutiérrez, and L. Muñoz, "Large-Scale Participatory Sensing Experimentation Using Smartphones within a Smart City," in Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. ICST, 2014, pp. 178–187.

[27] J. Paradells, C. Gomez, I. Demirkol, J. Oller, and M. Catalan, "Infrastructureless smart cities. Use cases and performance," in 2014 International Conference on Smart Communications in Network Technologies (SaCoNeT). IEEE, Jun. 2014, pp. 1–6.

[28] H. Sun, "Research of an intelligent street-lamp monitoring system based on the Internet of things," L. Zhang, L. Yu, and Y. Zhao, Eds., Mar. 2014, pp. 681–686.

[29] M. Granath and K. Axelsson, "Stakeholders' Views On ICT And Sustainable Development In An Urban Development Project," European Conference on Information Systems (ECIS), 2014, pp. 0–14.

[30] C. Shiyao, W. Ming, L. Chen, and R. Na, "The Research and Implement of the Intelligent Parking Reservation Management System Based on ZigBee Technology," in 2014 Sixth International Conference on Measuring Technology and Mechatronics Automation, no. 1. IEEE, Jan. 2014, pp. 741–744.

[31] E. Massung, D. Coyle, K. F. Cater, M. Jay, and C. Preist, "Using crowdsourcing to support pro-environmental community activism," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13. New York, New York, USA: ACM Press, 2013, p. 371.

[32] V. Gutiérrez et al., "SmartSantander: Internet of Things Research and Innovation through Citizen Participation," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013, vol. 7858 LNCS, pp. 173–186.

[33] D. Doran, S. Gokhale, and A. Dagnino, "Human sensing for smart cities," in Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13. New York, New York, USA: ACM Press, Aug. 2013, pp. 1323–1330.

[34] P. Marchetta et al., "S2-MOVE: Smart and Social Move," in 2012 Global Information Infrastructure and Networking Symposium (GIIS). IEEE, Dec. 2012, pp. 1–6.

[35] S. Roche and A. Rajabifard, "Sensing places' life to make city smarter," in Proceedings of the ACM SIGKDD International Workshop on Urban Computing - UrbComp '12. New York, New York, USA: ACM Press, Aug. 2012, p. 41.

[36] P. Mechant, I. Stevens, T. Evens, and P. Verdegem, "E-deliberation 2.0 for smart cities: a critical assessment of two 'idea generation' cases," International Journal of Electronic Governance, vol. 5, no. 1, 2012, p. 82.

 ISBN: 978-1-61208-438-1

[37] H.-N. Hsieh, C.-Y. Chou, C.-C. Chen, and Y.-Y. Chen, "The evaluating indices and promoting strategies of intelligent city in Taiwan," in 2011 International Conference on Multimedia Technology. IEEE, Jul. 2011, pp. 6704–6709.

[38] F. Gil-Castineira et al., "Experiences inside the Ubiquitous Oulu Smart City," Computer, vol. 44, no. 6, Jun. 2011, pp. 48–55.

[39] D. Havlik et al., "From Sensor to Observation Web with Environmental Enablers in the Future Internet," Sensors, vol. 11, no. 12, Mar. 2011, pp. 3874–3907.

[40] A. Botero and J. Saad-Sulonen, "Enhancing citizenship," in Proceedings of the 11th Biennial Participatory Design Conference on - PDC '10. New York, New York, USA: ACM Press, 2010, p. 81.

[41] J. Corchado, J. Bajo, D. Tapia, and A. Abraham, "Using Heterogeneous Wireless Sensor Networks in a Telemonitoring System for Healthcare," IEEE Transactions on Information Technology in Biomedicine, vol. 14, no. 2, Mar. 2010, pp. 234–240.

[42] K. Benouaret, R. Valliyur-Ramalingam, and F. Charoy, "CrowdSC: Building Smart Cities with Large-Scale Citizen Participation," IEEE Internet Computing, vol. 17, no. 6, Nov. 2013, pp. 57–63.

[43] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell, "A survey of mobile phone sensing," IEEE Communications Magazine, vol. 48, no. 9, Sep. 2010, pp. 140–150.

[44] A. S. Pentland, "Society's Nervous System: Building Effective Government, Energy, and Public Health Systems," Computer, vol. 45, no. 1, Jan. 2012, pp. 31–38.

[45] S. E. Middleton, L. Middleton, and S. Modafferi, "Real-Time Crisis Mapping of Natural Disasters Using Social Media," IEEE Intelligent Systems, vol. 29, no. 2, Mar. 2014, pp. 9–17.

[46] D. L. Estrin, "Participatory sensing," in Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10. New York, New York, USA: ACM Press, 2010, pp. 3–4.

[47] G. Cardone et al., "Fostering participaction in smart cities: a geo-social crowdsensing platform," IEEE Communications Magazine, vol. 51, no. 6, Jun. 2013, pp. 112–119.

[48] R. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," IEEE Communications Magazine, vol. 49, no. 11, Nov. 2011, pp. 32–39.

[49] M. C. Domingo, "An overview of the Internet of Things for people with disabilities," Journal of Network and Computer Applications, vol. 35, no. 2, Mar. 2012, pp. 584–596.

[50] M. F. Goodchild, "Citizens as sensors: The world of volunteered geography," GeoJournal, vol. 69, no. 4, 2007, pp. 211–221.

[51] M. N. Kamel Boulos et al., "Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, OGC standards and application examples," International Journal of Health Geographics, vol. 10, no. 1, 2011, p. 67.

[52] S. F. King and P. Brown, "Fix my street or else," in Proceedings of the 1st international conference on Theory and practice of electronic governance - ICEGOV '07. New York, New York, USA: ACM Press, 2007, p. 72.

[53] P. Meier, "New information technologies and their impact on the humanitarian sector," International Review of the Red Cross, vol. 93, no. 884, Dec. 2011, pp. 1239–1263.

[54] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele, "Participatory Air Pollution Monitoring Using Smartphones," Mobile Sensing: From Smartphones and Wearables to Big Data, 2012, pp. 1–5.

[55] E. DHondt, M. Stevens, and A. Jacobs, "Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring," Pervasive and Mobile Computing, vol. 9, no. 5, Oct. 2013, pp. 681–694.

[56] D. Richter, M. Vasardani, and L. Stirlng, Progress in Location-Based Services, ser. Lecture Notes in Geoinformation and Cartography, J. M. Krisp, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[57] H. Achrekar, A. Gandhe, R. Lazarus, Ssu-Hsin Yu, and B. Liu, "Predicting Flu Trends using Twitter data," in 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, Apr. 2011, pp. 702–707.

[58] S. Devarakonda et al., "Real-time air quality monitoring through mobile sensing in metropolitan areas," in Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing - UrbComp '13. New York, New York, USA: ACM Press, 2013, p. 1.

[59] M. Faulkner et al., "Demo abstract, the next big one: Detecting earthquakes and other rare events from community-based sensors," in Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks. Chicago: IEEE, 2011, pp. 13–24.

[60] G. Hancke, B. Silva, and G. Hancke, Jr., "The Role of Advanced Sensing in Smart Cities," Sensors, vol. 13, no. 1, Dec. 2012, pp. 393–425.

[61] T. Sakaki, M. Okazaki, and Y. Matsuo, "Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development," IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 4, Apr. 2013, pp. 919–931.

[62] E. Aramaki, S. Maskawa, and M. Morita, "Twitter Catches The Flu : Detecting Influenza Epidemics using Twitter The University of Tokyo," in Proceedings of the 2011 Conference on Emperical Methods in Natural Language Processing, Stroudsburg, 2011, pp. 1568–1576.

[63] H. Chourabi et al., "Understanding Smart Cities: An Integrative Framework," 2012 45th Hawaii International Conference on System Sciences, Jan. 2012, pp. 2289–2297.

 ISBN: 978-1-61208-438-1

A Knowledge Base for Electric Vehicles in Inner-City Logistics

Thomas M. Prinz, Johannes Kretzschmar, Paul Hempel, and Volkmar Schau

Chair of Software Technology
Friedrich Schiller University Jena, Germany
email: {Thomas.Prinz, Johannes.Kretzschmar, Paul.Hempel, Volkmar.Schau}@uni-jena.de

***Abstract*—Logistics companies depend on the new technology of electric vehicles when inner-city low emissions zones and their restrictions grow. The comprehensible utilization of electric vehicles in such a time and resource critical domain however requires an extensive software support regarding e-vehicle features. Since there are several logistics software systems, there is the need for a knowledge base for electric vehicles to allow a cross-application implementation of those features. In this paper, we argue for such a knowledge base and how it could basically look like. Furthermore, we motivate this base with some use cases. At the end, the paper closes with an exemplary knowledge base for the inference of possible drivers for a specific vehicle type.**

Keywords–Knowledge base; electric vehicles; logistics.

I. INTRODUCTION

The introduction of electric vehicles presents challenges for everyday life since they rise complete new technologies and handling. Otherwise, that introduction becomes more and more important as most big cities. Especially Germany and the Netherlands have low emissions zones restricting the type of vehicles. Inner-city logistics companies depend on that new technology to allow a supply in future as most conventional logistics vehicles have high emissions.

In our research project *Smart City Logistik Erfurt* (SCL) [1], we consider those challenges for the introduction of electric vehicles in inner-city logistics exemplary on the area of the city Erfurt. The major tasks are (1) a range forecast, (2) the driver's acceptance, (3) an open system architecture [2], and (4) a knowledge base:

(1) The range prediction is necessary to enable a precise tour planning since most tours in inner-city logistics should be planned in such that a vehicle uses its full range. Especially in the field of e-mobility, a solid capacity is required to prevent batteries from damage.

(2) The driver's acceptance is important as first tests have shown that the new technology, for example the range restriction of electric vehicles, makes drivers insecure. As result, a system has to support the drivers to school their handling to get a better time/costs ratio.

(3) Since the field of transport management systems offers less open application interfaces, there is the need for building an open system architecture to connect new systems for the consideration of electric vehicles (e.g., a range prediction) to current transport management systems. Before the introduction of electric vehicles can be successful in inner-city logistics, that interaction has to be implemented.

(4) Eventually, the knowledge base comprises necessary information of and behaviour rules for electric vehicles and inner-city logistics. Since there are currently a lot of software systems for logistics, such a base enables a cross-system implementation by system-independent terminologies, interdependencies, and inferences. It is therefore the base for all other mentioned topics and the content of this paper.

Traditionally, the development of knowledge-based systems consists of six steps (c.f. Figure 1): (1) identification, (2) conceptualization, (3) formalization, (4) implementation, (5) testing, and (6) revision [3]. Since there are diverse interdependencies between range influencing factors, we have to perform a *knowledge acquisition* as part of the identification step. Knowledge acquisition in the field of electric vehicles and inner-city logistics requires an analysis of the range influencing parameters, the business processes of logistics companies, and the participants as well as the resources in inner-city logistics, e.g., the structure of delivery tours.

For this purpose, we have to use several knowledge representations in our striven knowledge base. Detailed and structural descriptions of each resource, object, participant in inner-city logistics and in electric vehicles form the foundation. These descriptions define a controlled vocabulary [4] and follow a data-driven system approach [5]. Descriptions of numeric values specify units and their interdependencies, i.e., they allow for an automatic transformation from a source unit into a target unit. The structure of composed terms can be described with groups (compositions), cardinalities (arrays), optionalities, and polymorphisms [5]. Semantic annotations like synonyms, acronyms, textual information, and keywords allow for targeted searches and later comprehensive domain modelling.

Based on that *structural* layer of information, the next layer contains the interdependencies between the different information (or *concepts* in terms of ontologies). Those interdependencies define semantic information which allow for the inference of new or not explicit described information. For example, such a system can infer that a s-pedelec is subsumed by the concept of a moped.

Such a *conceptual* layer builds an advanced ontology for electric vehicles and inner-city logistics. However, the nature of description logics ontologies does not simply and efficient involve numerical interdependencies being the normal case in this field of research. If we consider the classes of European driving licences for example, then we see that one can receive only the driving licence class *B* if that person is at least 18 years old. To represent such a rule, there is the need for an additional layer — a *rule* base. The rule base includes the rules given by the ontology and additional numerical rules.

 ISBN: 978-1-61208-438-1

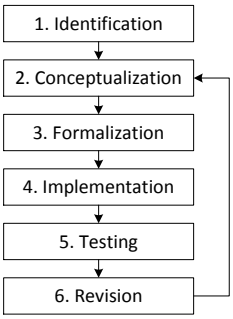


Figure 1. Expert system development after Buchanan et al. [3]

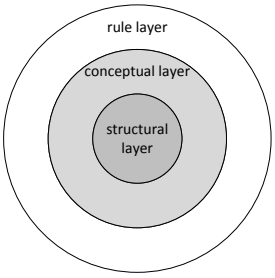


Figure 2. Approach for a knowledge base for electric vehicles

Our overall approach for a complete knowledge base for electric vehicles in inner-city logistics is illustrated in Figure 2. It contains the three above mentioned layers in concentric circles.

In this paper, we motivate some use cases for a knowledge base for electric vehicles in inner-city logistics at first (cf. Section II) and, afterwards in Section III, we consider the use case in European driving licence classes in more detail and demonstrate how we can use our three layer model to describe it. Finally, we close our paper with a short outlook into future work in Section IV.

II. Use Cases

The motivation for the construction of a knowledge base for electric vehicles in inner-city logistics is each of the following use cases, which arose from the SCL project: (1) Infer missing measurement data, (2) infer company important information, e.g., valid tours, drivers which are allowed to drive a specific car, drivers whose driving licence class expires, tour stops in valid time intervals, distances which are feasible for an electric vehicle of the company, checks of driver's rest periods, or goods with the same (or a close) destination and delivery time interval. Furthermore, a knowledge base may (3) provide mechanisms for actual and consistent data, e.g., current available electric vehicles.

In the remainder of this section, we consider these use cases in more detail with regard to their need and an idea for their solution with a knowledge base.

(1). Logistics software products and especially the range prediction use measurement data like global positions (GPS), current speeds, etc. for monitoring and optimization. However, the size of measurement data should be as small as possible without the loss of information. Sometimes, the system has to handle incomplete or contradictory data. For this reason, it (i.e., the knowledge base) has to be able inferring and evaluating the missing data. This is possible for data with physical correlations. For example, the average speed between two measurement points (i.e., two successive received measurement data) can be derived if the GPS positions and time stamps of both measurement points are given. Naturally, in some cases, there is a little loss of quality in the data since, for example, the distance which can be calculated with the help of two GPS positions may vary from the real distance.

Such an inference of measurement data is possible with the help of formal data descriptions, especially the physical correlations, which are defined in the structural layer of our proposed knowledge base.

(2). Logistics companies have the same trend to temporary workers, internationalization, globalisation, and optimization as other companies. For this reason, such companies are confronted with a wide heterogeneity of laws, structures, and cultural characteristics of different countries. It is necessary to collect all these (important) information to allow answers for simple questions whose inference is complex. As mentioned before, such questions could be: What is a good tour that is valid for a specific vehicle and fits all orders? Who of the drivers can drive that vehicle? Is it possible that a driver reaches each tour stop within a valid delivery time interval? When does a driver has to refresh its driving licence class to be continuously usable? Does all drivers observe the legal rest periods? Etc.

For such complex inferences, the structural layer must describe all concepts which belongs to driver licences, drivers, tours, goods, customers, vehicles, street maps, etc. Furthermore, the conceptual layer has to describe the relationships between those concepts and, eventually, the rule layer defines additional rules for those relationships. How such a knowledge base could be implemented is shown exemplary in the next section on the question "Who of the drivers can drive that vehicle?".

(3). In this fast-moving time, it is important to keep up-to-date. If the knowledge base uses standardized data descriptions of concepts, it can automatically support eventual update processes, by checking for consistency between comprehensive domain ontologies. Often, new electric vehicles have a better power performance and, therefore, save time and money. For this reason, the structural, the conceptual and the rule layer have to use standardized data formats or should be involved in standardizations.

These use cases show that a knowledge base, which provides more functionality as a simple data base is useful in the context of logistics software and electric vehicles. In the next section, we present a cut-out of a possible knowledge base for European driving licence classes.

III. A Knowledge Base for European Driving Licence Classes

In this section, we exemplary introduce (parts of) a knowledge base for driving licence classes in the European Union. The use-case for this knowledge base is to derive drivers who can drive a specific vehicle type or vehicle types who can be driven by a specific driver.

In the European Community, a lot of driving licence classes exist. Figure 4 shows these classes and also a cut-out of their interdependencies. As we see, there are multiple

 ISBN: 978-1-61208-438-1

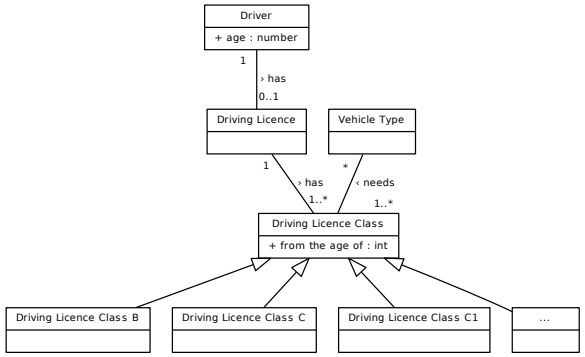


Figure 3. Entities in the knowledge base

interdependencies between those driving licence classes, which are difficult to know and to learn for an inexperienced user. For this reason, a knowledge base would be profitable for supporting logistic scheduler.

As argued in Section I, at first, we introduce the structural parts of our striven knowledge base. This structural layer consists of data descriptions, i.e., the description of the concepts. In our implementation, we have used an own data description language, which structures these concepts in groups, cardinalities, options, and entities as introduced in Döbrich and Heidel [5]. Those data descriptions contain also synonyms and textual descriptions as well as units. For a better readability, we use an UML 2.0 class diagram [6] at this point of view. That class diagram is illustrated in Figure 3.

The major concepts are the ones to represent a driver, a driving licence, a vehicle type, and driving licence classes. Naturally, several other concepts can be introduced to describe those concepts in more detail.

Furthermore, there are some connected attributes for these concepts. As start point, the concept *driver* consists of an age. A driver has up to one *driving licence*, which has at least one *driving licence class*. Such a class has a class-specific driver's age for which that class can be received of a person. Furthermore, a driving licence class is needed to drive several *vehicle types*. At last, there are some subtypes of driving licence classes like *driving licence class B*.

After we have build the structural layer, we have to introduce the conceptual layer. As mentioned in Section I, the conceptual layer contains relations between these concepts. Some of these relations are already defined in the structural layer. At first, for each class in the class diagram, we include an unary relation in our knowledge base. For this, in the following, we use (to represent arbitrary instances) variables d for representing a driver, c, c_1, c_2 for driving licence classes, a, a' for ages, dl for a driving licence, and finally v as an arbitrary instance of a vehicle type:

$$\begin{aligned} Driver(d)\ (= D(d)) \\ Driving\_Licence(dl)\ (= DL(dl)) \\ Driving\_Licence\_Class(c)\ (= DLC(c)) \\ Driving\_Licence\_Class\_C(c)\ (= DLC\_C(c)) \\ Driving\_Licence\_Class\_C1(c)\ (= DLC\_C1(c)) \\ Driving\_Licence\_Class\_B(c)\ (= DLC\_B(c)) \\ Vehicle\_Type(v)\ (= VT(v)) \end{aligned}$$

Afterwards, we need the explicit binary relations in the following equation, which are extracted from the associations and the attributes of the class diagram:

$$\begin{aligned} From\_The\_Age\_Of(c, a)\ (= FTAO(c, a)) \\ Age(d, a) \\ Has\_Driving\_Licence(d, dl)\ (= HasDL(d, dl)) \\ Has\_Driving\_Licence\_Class(dl, c)\ (= HasDLC(dl, c)) \\ Needs(v, c) \end{aligned}$$

Eventually, we introduce the subclass-associations of the class diagram as rules into our conceptual layer:

$$\begin{aligned} DLC\_C(c) \rightarrow DLC(c) \\ DLC\_C1(c) \rightarrow DLC(c) \\ DLC\_B(c) \rightarrow DLC(c) \end{aligned}$$

Now, we have a stable structural and conceptual layer for our knowledge base (for this cut-out). Like Figure 4 shows, there are many other interdependencies between driving licence classes and the involved concepts. To represent those dependencies, we have to create a rule layer, which contains additional information.

As basic for our rule layer, we want to check whether an arbitrary number ($x \in \mathbb{R}$) is greater than or equal to another number ($y \in \mathbb{R}$):

$$GEq(x, y) = "x \geq y"$$

Whether two driving licence class instances belong to the same driving licence class can be checked by $Equal$:

$$\begin{aligned} DLC\_C(c_1) \wedge DLC\_C(c_2) \rightarrow Equal(c_1, c_2) \\ DLC\_C1(c_1) \wedge DLC\_C1(c_2) \rightarrow Equal(c_1, c_2) \\ DLC\_B(c_1) \wedge DLC\_B(c_2) \rightarrow Equal(c_1, c_2) \end{aligned}$$

As shown in Figure 4, a driving licence class C includes the driving licence class C1, i.e., one driver having a class C can also drive vehicles, which needs class C1.

$$DLC\_C(c_1) \wedge DLC\_C1(c_2) \rightarrow Includes(c_1, c_2)$$

Furthermore, the same figure shows, that both classes C and C1 requires class B to be received. Thus, each driver with class C, for example, can also drive vehicles with class B.

$$\begin{aligned} DLC\_C(c_1) \wedge DLC\_B(c_2) \rightarrow Requires(c_1, c_2) \\ DLC\_C1(c_1) \wedge DLC\_B(c_2) \rightarrow Requires(c_1, c_2) \end{aligned}$$

Now, we can infer all driving licence classes, which are available with a single one when we merge the $Equal$, $Include$, and $Requires$ rule to a single $Contains$ rule:

$$\begin{aligned} Equal(c_1, c_2) \rightarrow Contains(c_1, c_2) \\ Includes(c_1, c_2) \rightarrow Contains(c_1, c_2) \\ Requires(c_1, c_2) \rightarrow Contains(c_1, c_2) \end{aligned}$$

As a kind of validation, we check whether a specific age is enough to allow the receiving of a driving licence class. Therefore, we overload the rule $Requires$:

$$DLC(c) \wedge FTAO(c, a') \wedge GEq(a, a') \rightarrow Requires(c, a)$$

To build a predicate that allows us the inference of the drivers who can drive a specific vehicle type, we have to derive the driving licence classes of a driver.

$$HasDL(d, dl) \wedge HasDLC(dl, c) \rightarrow HasDLC(d, c)$$

 ISBN: 978-1-61208-438-1

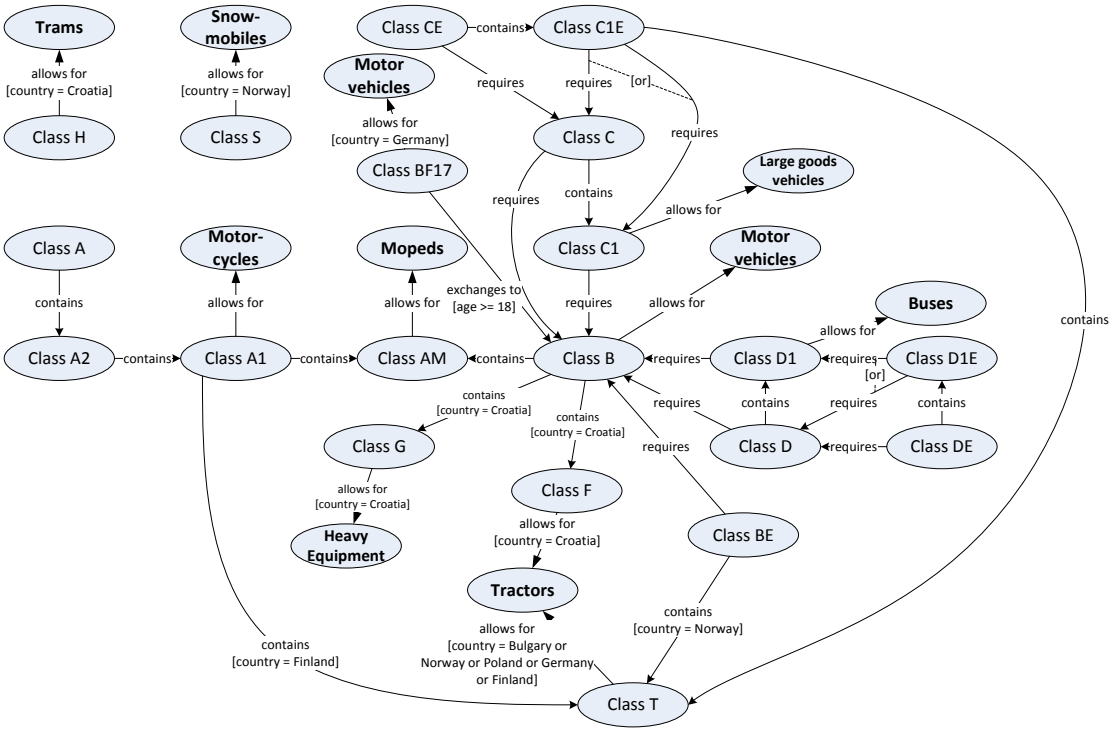


Figure 4. European driving licence classes and their interdependencies

To be sure that a driving licence class is valid for a driver, we introduce a rule that checks the age of the driver with regard to a specific driving licence class:

$$D(d) \wedge DLC(c) \wedge HasDLC(d,c) \wedge Age(d,a) \\ \wedge Requires(c,a) \rightarrow DHasDLC(d,c)$$

Finally, we can create a rule for our rule layer that allows us to infer all vehicle types that can be driven by a driver and all drivers that are allowed to drive a specific vehicle type:

$$VT(v) \wedge Needs(v,c_1) \wedge DHasDLC(d,c_2) \wedge Contains(c_2,c_1) \\ \rightarrow DCanDriveV(d,v)$$

With a simple implementation of our knowledge base, e.g., in Prolog, we can infer our wanted information by replacing a specific driver or vehicle with a variable, i.e., a place holder. That functionality can be adapted to logistics software, which automatically infer missing driving licence classes in documents and possible drivers for a tour. So, a knowledge base for electric vehicles is justifiable.

IV. Conclusion

In this work of progress paper, we have argued for a knowledge base for electric vehicles in inner-city logistics. Therefore, we have divided our knowledge base into three layers: the structural, conceptual, and rule layer. Furthermore, we have identified and explained some use cases for such a knowledge base and, finally, showed that it is possible to create such a knowledge base on an exemplary use case, which infers drivers, who are allowed to drive a specific vehicle type.

With the help of such a knowledge base and the presented use cases, it is possible to implement an intelligent software user interface for logistics that helps to create a consistent and complete data base. For example, if a company recruits a driver, the system can infer all of its implied driver licence classes although the driver stated the superordinate class $C1$.

In the future work, we have to extend our knowledge base with additional concepts, relations, and rules to allow more use cases and more safe inferences. Therefore, more knowledge has to be derived from the practice of using electric vehicles.

V. Acknowledgement

The project is funded by the German Federal Ministry for Economic Affairs and Energy, BMWi.

References

[1] V. Schau et al., “SmartCityLogistik (SCL) Erfurt: Deriving the main factors that influence vehicle range during short-distance freight transport when using fully electric vehicles,” in 10. GI/KuVS-Fachgespräch ”Ortsbezogene Anwendungen und Dienste”, pp. 101–108.

[2] S. Apel, T. M. Prinz, and V. Schau, “Challenging service extensions for electric vehicles in massively heterogenic system landscapes,” in Proceedings of the 7th Central European Workshop on Services and their Composition, ZEUS 2015, Jena, Germany, February 19-20, 2015., pp. 44–50.

[3] B. G. Buchanan et al., “Constructing an expert system,” Building expert systems, vol. 50, 1983, pp. 127–167.

[4] N. I. S. Organization, ANSI/NISO Z39.19 - Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies, N. I. S. Organization, Ed. National Information Standards Organization, May 2010, iSBN 978-1-880124-65-9.

[5] U. Döbrich and R. Heidel, “Datadriven Program Systems - a way out from interface chaos.” Informatik Spektrum, vol. 35, no. 3, 2012, pp. 190–203.

[6] J. Rumbaugh, I. Jacobson, and G. Booch, Unified Modeling Language Reference Manual, The (2nd Edition). Pearson Higher Education, 2004.

 ISBN: 978-1-61208-438-1

Building a Service Manager For a Smart City Archicture

Towards a service manager in an interoperable environment

Gutemberg Rodrigues Costa Cavalcante[1], Felipe Silva Ferraz[1,2], Guilherme Luiz Mario de Medeiros[1]

[1]CESAR
Recife Center for Advanced Studies and Systems
Recife, Brazil
gutembergrcc@gmail.com
fsf@cesar.org.br
guicaraciolo@gmail.com

[2]Informatics Center
Federal University of Pernambuco
Recife, Brazil
fsf3@cin.ufpe.br

Abstract - **Cities are becoming more and more populous and complex, and this growth is forcing them to better administer their management services. As a result of this growth, and of technological advances, cities are investing in technology so as to become smarter, thereby obtaining quicker results. This technological scenario has not only produced benefits for cities but also fragilities in them. Since the services that a city offers are vital and some of these require confidentiality, the focus has shifted to information security. To ensure their information is covered, cities need specific technologies, such as City Security Layer (CSL), in order to solve security problems arising. This paper focuses on constructing a module that complements CLS. This module is responsible for managing the services available in a network controlled by CSL.**

Keywords- security; smart city; architecture; services.

I. INTRODUCTION

Cities are constantly growing. Nam et al. [1] assert that they are becoming more and more populous and complex. According to Dirks et al. [2], in the 20th century, less than 20 cities around the globe had more than one million citizens. Today, this number has risen to 450 cities. Given this demographic growth, cities are encountering new series of risks, concerns and problems. According to Nam et al. [1], the main problems will be: a deterioration in the quality of the air, in traffic flows and an increase in economic risks, such as greater unemployment and the challenge of ensuring the best use of communication technologies so that it is possible to offer citizens an infrastructure that will become more and more prosperous [3][4].

With regard to the prosperity of cities, according to Sen et al. [5], this could be achieved when the ways that people think about health, security and economic issues are as important as their thinking on tackling uncontrolled urban development. According to Dirks et al. [2], to attend to these matters, the main services that cities offer should become interconnected, thus enabling new intelligence levels to be attained and, therefore, able to meet their own demands and those of their citizens.

In these cities, what is perceived is not only population growth, but also, as Dirks et al. [2] point out, such cities undergo a rise in their economic and technological activities. On the other hand, for Sen et al. [5], it is important to state that revolutionary change in communications is imminent. Such breakthroughs are becoming a reality and arise out of the services being created in cities.

Moreover, according to Sen et al. [5], the option to create services forces and makes software programs even smarter and more and more connected, to such an extent that they can exchange information, thus allowing new solutions to be created. The vision for smart cities is to see them as interconnected urban areas [6][7], which are sustainable and efficient, since all city services are crafted and maintained by focusing on their sharing data with each other. Therefore, it is possible for cities to gather information and take decisions more quickly and reliably. This integration of and between city services is not only a source of benefits, but it also is an imminent point of problems or vulnerabilities when information security is taken into account [8]. This is why Bartoli et al. [8] affirms that one of the biggest challenges when developing smart cities is related to the security of systems.

According to Bartoli et al. [8], Information Security should not only deal with deliberate attacks, such as those by disgruntled employees or for the purposes of industrial espionage, but also vulnerabilities such as that from a malicious entity that has penetrated a network [5][8], and thus has access to how software and data are controlled and, therefore, it can modify and damage the entire system.

This study was prompted after noting the lack of research studies on information security concepts with regard to the peculiarities of urban environments or smart cities. Among the few published papers, CSL stands out in the management of identifiers of entities but there remains the need to extend this solution to include the register of services that a smart city will consist of.

This article is organized as follows: Section 2 addresses how to define a smart city and the different services it may offer. Section 3 defines the security challenges that smart cities need to face up to. Section 4 discusses the CSL security layer, how it is structured, and what challenges it tries to overcome. Section 5 describes the *Service Manager* as a solution for managing services of a smart city. Section 6 sets out a validation of the *Service Manager* module with the CSL layer. Finally, some conclusions are drawn and suggestions made for future research studies in Section 7.

 ISBN: 978-1-61208-438-1

II. SMART CITY AND SERVICE DISTRIBUTION

According to Dirks et al. [2], rapid growth in population creates a new set of challenges for the infrastructure services of cities while, at the same time, creating new economic opportunities and social benefits.

Washburn et al. [9] is of the opinion that as people migrate to urban areas, resources become limited and badly managed. As a result of this, Dirks et al. [2] point out that problems will arise to do with high costs of living. For example, as to fresh water, it is expected that it will increase 25% in price by the mid-2030s. The high cost of living in some cities can already be observed in terms of people looking to the private sector due to the lack of some basic provisions in health and education services by government.

For Dirks et al. [2], cities that are already facing these challenges need to act by making using of new technologies so as to transform their systems and, in so doing, they will be better able to manage their resources and thus become more competitive. In order to achieve this, Ferraz et al. [7] and Kanter et al. [10] state that, when the tools and services of cities are integrated into a network, they will contribute to higher efficiency, since they will be able to use an enormous range of information, thereby enabling them to be creative and to make assertive decisions that are well-founded.

A city has different systems and distributed services. What we understand as services and systems is the combination of the complete range of resources for a specific function. Such services may be represented as being part of a set, which according to Ferraz et al. [7] can be separated and organized into several categories: education, public safety, transport, energy and water, health and governmental bodies.

According to Dirks et al. [2], we should note that what these services comprise, may vary from city to city, and in the number of citizens, since each city has its own characteristics, but nevertheless within the groups presented and defined. Dirks et al. [2] and Ferraz et al. [7], go on to state and demonstrate that the effectiveness and efficiency of these services will determine how successful the city that provides them will be. In the next section, each category of service in a smart city will be analyzed.

A. Types of Service

A city can offer different types of services. According to Ferraz et al. [7], services can be in the following areas: education, public safety, forms of transport, government services, health, energy and water.

Education: This represents the services that are directly and indirectly related to all educational services, such as, for example, setting standards for student's grades or educational shills.

Public Safety: This represents the services that help cities to respond quickly responds to emergencies, thus guaranteeing safety in a city. With the help of these services, for example, we are able to identify the rate of theft in certain areas.

Transport: Transport services include the state of roads, seaports, and airports. For example, controlling the volume and flow of traffic on city roads.

Government: This represents each system which works within governmental frameworks. For example, the control of a city's budget and expenditure.

Health: This represents services that help improve public health. By using these services, users will be able to have a shared medical record, that is always available, and which will lead to quicker and more precise diagnoses.

The smart integration of those services in a smart city will not only deliver benefits. One example is the evolution of the health services, where paramedics or even patients can be advised at a distance how to store and apply drugs. For Verbauwhede [11], this evolution will only be possible when there are strong privacy and user authentication policies. This privacy and authentication can be supplied by providing artefacts with protocols and cryptographed application software, as will be seen in the following section.

III. SECURITY IN SMART CITIES

According to Bodei et al. [12], studies showed the need for a new set of research studies focusing on improving information security, when dealing with smart cities [5][8].

In the midst of the problems related to information security, a subset of security questions is present in the backdrop to smart cities, amongst which worthy of special mention are access to information, tracking items of information and citizens, loss of data and unauthorized access to datacenters.

The issues above are dealt with in a broader study undertaken by Ferraz et al. [7], and are presented here as a way to illustrate points that smart cities will need to address.

These three issues are discussed in the following subsections.

1) Issues related to access to information

The interaction between software and the network involves data sharing. According to Sen et al. [5], this interaction can represent a threat since the data from different entities can become accessible. The traffic of packets from a device to the network, and from the network to other devices is a concern, since these packets can be intercepted when they are being transferred.

2) Issues related to data tracking

One important characteristic in an interconnected environment is the fact that a set of information used by one system cannot be traced back to the originator of the data. This kind of problem can destroy the anonymity that the services supply.

3) Issue related to entity tracking related issues

Each smart city may have many distributed sensors to capture data and facilitate the integration of systems. For Sen

 ISBN: 978-1-61208-438-1

et al. [5], information from these sensors must not be used to track entities. More information can be found in [13][14].

All problems here described are related to access and security matters. For Bartoli et al. [8], an alternative for solving some of those problems is by using key management. This means providing secure management of data encryption. The author states such management will ensure users are authenticated and authorized. According to Bartoli et al. [8] and Li et al. [15], for effective protection in a smart city, a series of security related problems needs to be addressed, and a predefined plan or goal should be adhered to.

What this consists of will be discussed in the next section. Similar solutions will be analyzed and a new layer will be put forward that aims to solve the problem of controlling entities and ensuring authenticity in smart cities.

IV. CSL

This section will discuss the CSL approach, as a solution to identity security in a smart city. In the first topic, a brief description of the problem will be given. The second topic will present similar solutions and the third and final topic will discuss the CSL.

A. Problem

In Section 2, the concept of a smart city was described as comprising different services. Given the growing number of such services and related entities, the complexity of security problems has also kept growing.

Among the problems detected, attention is drawn to the situation when the information service requests the service provider to supply it with confidential personal information on third parties.

According to Ferraz et al. [14], exchanging information via a network is considered to be unreliable because messages are subject to losses and interception when they are transmitted, as set out under security in Section 3. In short, the problem identified is based on guaranteeing the anonymity of an entity within the environment of a smart city.

There are already some solutions on the market that address these problems but none of them focuses on smart cities, as will be shown in the next topic.

B. Similar solutions

Some of the problems mentioned in Section 3 can be mitigated by some of the existing security solutions. According to Ferraz et al. [17] approaches such as using Open Authorization (OAuth), Security Assertion Markup Language (SAML) and OpenID may help the process of giving cover to some security flaws.

OAuth (Open Authorization), according to Yang et al. [18], is an authentication protocol used for storing secure data, whereby the owner of the storage does not need to provide his access credentials. The Security Assertion Markup Language (SAML), according to Saklikar et al. [19], is an XML-based framework for exchanging authentications, authorizations and data. By using SAML, a relationship of trust between entities in a network environment can be created. On the other hand, OpenID is an open technology in which, according to Ferraz et al. [17], users are identified by a URL. In systems using OpenID, users do not need to create a new account to access them. Users only need to be authenticated by an identity provider.

For Ferraz et al. [17], solutions like OpenID, SAML and OAuth, are fundamental to ensure users' security. However, these protocols cannot cover all existing security problems. Most of these concerns are related to the fact those solutions are focused on authentication and authorization, which, in an environment full of sensitive data, is not sufficient. CSL arose with a view to having a solution for dealing with anonymity between entities in a smart city and will be described in the next topic.

C. CSL Solution

CSL represents a layer that should be placed on the external border of a service, or a set of services, in smart cities.

According to Ferraz et al. [20], the main objective of CSL is to be the layer responsible for modifying the identifiers of services when messages are exchanged. The new identifier will be generated based on combining the previous identifier with the service to be accessed.

For Ferraz et al. [20], by means of this process, it is possible to ensure that an entity keeps itself anonymous within the entire smart city environment, even when this entity accesses multiple services, since the creation of the identifier consists of combining two other identifiers. The final access identifier will be different for each of the services.

By using CSL, it is expected that there will be an increase in security, since there will now be a layer which will provide a unique identifier for each service. With this approach, the real identity of the entity is preserved. On the other hand, Ferraz et al. [20] stresses that CSL does not provide resources to ensure authentication and authorization, which OAuth, SAML and OpenID do. Therefore, making use of an extra layer or solution to address vulnerabilities on this matter is required.

Despite the limitations mentioned, the security problems shown in Section 3 are partially covered by CSL. Table 1 illustrates the coverage of each item.

TABLE I. CSL COVERAGE

Risks	Coverage
1. Data Access	✓
2. Data tracking	✓
3. Entity tracking	✓

According to Ferraz et al. [20], if any packets shared between entities which go through CSL are intercepted, it will be hard to identify and understand who is who, since the interceptor will not know what the identifiers of these entities are. This covers the first issue, described in Section 3.

 ISBN: 978-1-61208-438-1

When there is anonymity of the entities during communication, the second issue is covered as well, since the difficulty of identifying which information is associated with which entity makes it hard to map the relationships. This characteristic also covers the third issue, since the entities are isolated.

The proposed approach has presented an efficient solution for some of the issues mentioned. Also, another set of issues is partially addressed by **C**ity **S**ecurity Layer (CSL).

During the development of CSL, as presented in The First International Conference on Advances and Trends in Software Engineering in Barcelona – Spain, the need for a service manager to manage connected services was raised. The next section describes the specification and definition of that component.

V. SERVICE MANAGER

After having understood the CSL, it is seen that there is a demand for the Service Manager module. The module is integrated above the security solution by orchestrating the management of services. This section will detail the module by describing its architecture and its functionalities.

A. Motivation

The Service Manager is a module for managing and controlling services that are present in a smart city. This module is characterized as a plug-in which, when added to the CSL, will be responsible for controlling its services.

The need to have the service manager arises because the CSL need to know all the services that can communicate with each other in a smart city. The need for registering services is met by obtaining and filling in the CSL hash table so that organizations know who is available for them to communicate with.

Figure 1 shows two levels of security depth where the CSL and Service Manager are present.

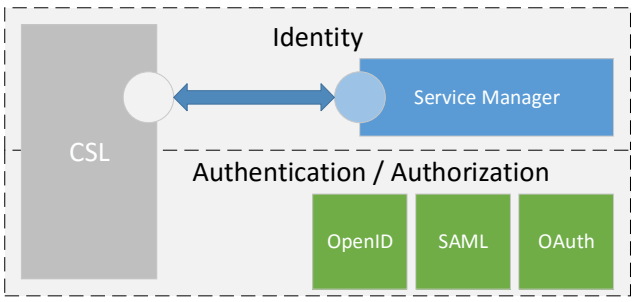


Figure 1. Summary of the ideal security environment

OAuth, SAML and OpenID appear as similar solutions that are on the same level of security of the CSL. After the two levels of security, there are network protocols that helped complement the solution.

B. API

The API provided by the Service Manager consists of the entry points provided by the CSL. These are:

Registration of Services: This allows the insertion of services that are part of the interoperable systems and solutions of a city. For registration, the service information that needs to be sent includes: name, description, public identifier and the URL address. On entering the service, negotiable validations are made to avoid replication of the same services and mandatory data. If the registration of the service satisfies the validation, the service is registered and returned to whoever requested the registration.

Alterations of Services: Editing information of the registered services. The alteration is carried out with validations to avoid the inconsistency in the data of the service. This precaution is taken because some data of the service comprise the Table of identifiers that the CSL uses to route information to the services.

Removal of Services: Exclusion of services that do not take part in the environment of the city. This is only authorized if the service has not taken part in any interaction in the city, otherwise it should be disabled.

Listing of Services: Listing the information that make up the services, such as: name, description, address of the service and public id. The service assembles a dynamic table of the services apt to play a part in an interaction in the city.

Loading of Services: The function guarantees the possibility of migrations of the services from a city to the CSL. To load the services, a file in text format will be requested, in which a standard must be respected: {name, description, public id, URL address}, {name, description, public id, URL} ... When the file is loaded, the services present in it will be inserted if they respect the inclusion rules.

Address Resolver: This provides the URL address of the service being requested to in order to send on the piece of data of whoever asked for it to the party requested.

C. Look and Feel

Figure 2 shows the main screen responsible for maintenance services. On this screen, the user can manage each service used by the CSL.

Figure 2. Main screen maintenance services

Figure 3 presents the screen related to editing a registered service.

Figure 3. Editing registered Services

These two screens refer to the core parts of service management. They also they play an important role in the general functionality of CSL.

 ISBN: 978-1-61208-438-1

D. The Workflow of Service Manager

The workflow between CSL and the Service Manager is described in Figure 2:

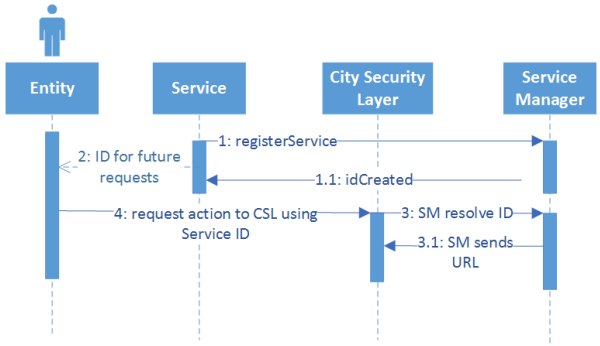


Figure 4. Sequence diagram with the workflow between CSL and Service Manager

Description:

1. A service is registered on Service Manager;
1.1. The Service Manager returns the identifier to the new service;
2. This identifier will be used for future requisitions;
3. CSL, by using the identifier of the required service, makes requests to the Service Manager at the URL for this service.
4. The entity will keep on requesting service information by sending messages through CSL.

E. Technologies Used

The programming language in which the solution was structured was Java. The choice was based on the fact that the CSL was originally built with it, thus facilitating its integration and because it has a large number of communities which aim to facilitate development work by constructing frameworks [21].

In the presentation layer of the Service Manager, Java Server Faces (JSF) version 2.2 was used. JSF is a technology which permits Java for Web applications to be created using ready-made visual components so that the developer only has to be concerned about its use [22]. Together with the JSF, Primefaces was adopted. According to [23], this extension of the JSF stands out due to its simplicity, performance and template options.

The other technologies used in the CSL were not altered so that the solution of the manager does not have an impact on the existing security layer. For transactional control, the CLS uses the Spring framework [24] which besides assisting communication with the other layers aims to remove the dependencies between entities with the injection of dependency. In the data layer, use is made of the JPA framework [25] which will orchestrate the transactions with the database and avoid code repetition when dealing with the persistence of data. The view of the layers can be seen below in Figure 5.

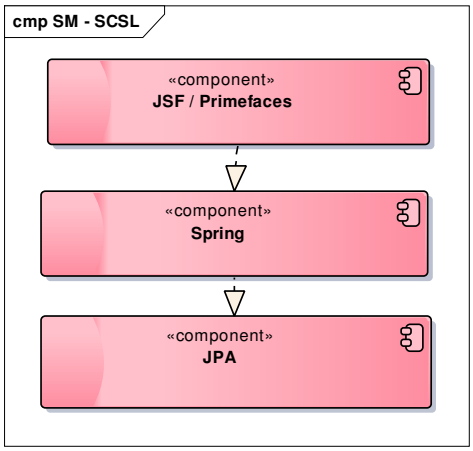


Figure 5. Technologies used

The Java programming language contributed to the creation of a modular environment and with independent layers, whereby one layer provides a service to the one above it. The CSL provides basic services to the Service Manager, and delegates responsibility for validation, treatment and manipulation of services.

The JSF helped in dealing with requests for communication with the service layer of the CSL and dealing with events. Primefaces was used to provide graphical interface components, such as templates, buttons, and dynamic tables.

VI. VALIDATION

The validation process will evaluate the Service Manager incorporated into the CSL. To do so, it will count the time spent on including, excluding, and requesting services on CSL, with and without using this management module. By using this measure, a stress test will be exclusively used to validate performance.

A. The infrastructure for measurement

The validation environment was executed on an Intel Core I5 computer, with 8GB memory RAM, which uses the Windows 8 operating system.

B. Unity and count tool

The unit of measurement was milliseconds (ms). To draw up the validation test for inclusions in, exclusions from and requests to the CSL, the JMeter tool was used, which is a free, open source tool by the Apache Foundation and used to test the performance of software applications.

The metrics defined is a time-count of conducting 1000 samples of including, excluding, and consulting services. This activity will consider the presence, or absence, of the Service Manager, and will record a comparison of time-counts before and after using the manager in the CSL solution.

 ISBN: 978-1-61208-438-1

C. Results and Analysis

The results obtained after applying the metrics of the previous topic are shown in Tables II and III.

TABLE II. VALIDATION RESULT

Operation	Median	
	With Service Manager	without Service Manager
Include	4.000	3.850
Remove	2.9000	2.500
Query	3.800	3.600

TABLE III. VALIDATION RESULT IN PERCENTAGE

Operation	Percentage Increase
Include	3.75 %
Remove	13.79%
Query	5.26%

As seen in the above tables, the addition of the Service Manager, even when there is an interface and control layer did not cause a significant increase in the basic operations of addition, deletion and consultation of services.

VII. CONCLUSION

Cities grow constantly. This is caused by people migrating to urban areas, or the growth of their own populations. This growth is forcing cities to organize themselves better and continuously, since people are demanding even more resources and consuming even more services. It is also obliging cities to invest in information technologies, and to start to become smarter. The new technologies are helping cities to obtain faster results and to attend to the demands of their citizens.

By using these technologies, cities are starting to be called smart cities. Nevertheless, this new technological scenario has led to such cities having to face a set of fragilities. For example, since the services that cities provide are extremely vital, and some of these may require data to remain confidential, the new focus is now on security in the smart city.

Nowadays, there is a set of solutions that partially covers these fragilities, as seen in this study, namely OAuth, SAML and OpenID. Even though these solutions are focused on authentication and authorization, throughout this article, it was demonstrated that the issue of identity control in the smart city has not been adequately addressed. It was as a solution for this identity issue that CSL was conceived as a layer that would be responsible for ensuring anonymous communication between entities and services, thereby covering some of the security problems of smart cities.

As a way of complementing CSL, this article puts forward a structure for creating a service manager. As demonstrated, CSL needs to know all the services that will be available for communication purposes in a smart city, since CSL needs to maintain a table of identifiers.

Because of these needs, a service management module was developed for CSL. To validate this module, a scenario was built in which to test the performance of including, listing, excluding services and requesting these services, with and without this new module.

On analyzing the results, it was shown that adding a new module to compare the time spent on each type of operation leads to barely increasing the time spent on management, thus showing this new module when plugged in to CSL is viable.

For future studies, more research on CSL will be needed as CSL evolves, since its current scope only contemplates a small set of security concerns. In addition, a future study should contemplate using CSL with the service manager module in a more complex smart city, to enable further validations and metrics.

REFERENCES

[1] T. Nam and T. A. Pardo, “Conceptualizing smart city with dimensions of technology, people, and institutions,” Proc. 12th Annu. Int. Digit. Gov. Res. Conf. Digit. Gov. Innov. Challenging Times - dg.o ’11, 2011, p. 282.

[2] S. Dirks and M. Keeling, “A vision of smarter cities: How cities can lead the way into a prosperous and sustainable future,” IBM Inst. Bus. Value. June, 2009.

[3] F. Duarte, “Smart Cities: technological innovation in urban areas”, São Paulo em Perspect., vol. 19, 2005, pp. 122–131.

[4] J. Shapiro, “Smart cities: quality of life, productivity, and the growth effects of human capital,” Rev. Econ. Stat., vol. v88(2,May), 2006, pp. 324–335.

[5] M. Sen, A. Dutt, S. Agarwal, and A. Nath, “Issues of Privacy and Security in the Role of Software in Smart Cities,” 2013 Int. Conf. Commun. Syst. Netw. Technol., 2013, pp. 518–523.

[6] Global Electronics Industry, “The IBM vision of a smarter home enabled by cloud technology,” 2010, p. 16.

[7] F. Ferraz, C. Sampaio, and C. Ferraz, “Towards a Smart City Security Model Exploring Smart Cities Elements Based on Nowadays Solutions,” ICSEA 2013, 2013, pp. 546–550.

[8] A. Bartoli, M. Soriano, J. Hernandez-Serrano, M. Dohler, A. Kountouris, D. Barthel, Security and Privacy in your Smart City , in Proceedings of Barcelona Smart Cities Congress 2011, 29-2 December 2011, Barcelona (Spain).

[9] D. Washburn, U. Sindhu, and S. Balaouras, “Helping CIOs Understand ‘Smart City’ Initiatives,” Forrester, 2009.

[10] R. M. Kanter and S. S. Litow, “Informed and Interconnected : A Manifesto for Smarter Cities Informed and Interconnected : A Manifesto for Smarter Cities,” Working Paper 09-141, Havard Business School, 2009.

[11] I. Verbauwhede, “Efficient and secure hardware, for cryptographic algorithms on embedded devices,” 2012, pp. 1–4.

[12] C. Bodei, P. Degano, and G. L. Ferrari, “Formalising security in ubiquitous and cloud scenarios,” In Proc. 11th IFIP TC 8 International Conference on Computer Information Systems and Industrial Management, 2012, pp. 1-29.

[13] F. S. Ferraz and C. A. G. Ferraz, “More Than Meets the Eye In Smart City Information Security: Exploring security issues far beyond privacy concerns,” in IEEE computer science, UFirst-UIC 2014, 2014, pp. 677-685.

[14] F. S. Ferraz and C. A. G. Ferraz, “Smart City Security Issues: Depicting Information Security Issues in the Role of an Urban Environment,” in 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014, pp. 842–847.

 ISBN: 978-1-61208-438-1

[15] W. Li, J. Chao, and Z. Ping, “Security Structure Study of City Management Platform Based on Cloud Computing under the Conception of Smart City,” 2012 Fourth Int. Conf. Multimed. Inf. Netw. Secur., 2013, pp. 91–94.

[16] F. J. L. Ribeiro, J. C. R. Lopes, and A. C. P. Pedroza, “Analysis of security processes in mobile third generation systems,” I Escola Regional de Redes de Computadores, Porto Alegre, Brazil, September 2003.

[17] F. S. Ferraz, C. Candido, B. Sampaio, C. André, and G. Ferraz, “Information Security in Smart Cities Using OpenID , SAML and OAuth to increase security in urban environment,” SOFTENG 2015 First Int. Conf. Adv. Trends Softw. Eng., 2015, pp. 7–13.

[18] F. Yang and S. Manoharan, “A security analysis of the OAuth protocol,” IEEE Pacific RIM Conf. Commun. Comput. Signal Process. - Proc., 2013, pp. 271–276.

[19] S. Saklikar, S. Saklikar, S. Saha, and S. Saha, “Next steps for security assertion markup language (saml),” Proc. 2007 ACM Work. Secur. web Serv., 2007, p. 65.

[20] F. S. Ferraz, C. Candido, B. Sampaio, C. André, and G. Ferraz, “Towards A Smart-City Security Architecture Proposal and Analysis of Impact of Major Smart-City Security Issues,” SOFTENG 2015 First Int. Conf. Adv. Trends Softw. Eng., 2015, pp. 108–114.

[21] “Java.” [Online]. Available: https://java.com/pt_BR/about/whatis_java.jsp. [Accessed: 20-Oct-2015].

[22] “JavaServer Faces.” [Online]. Available: http://docs.oracle.com/javaee/5/tutorial/doc/bnaph.html. [Accessed: 28-Oct-2015].

[23] “Primafaces.” [Online]. Available: http://www.primefaces.org/whyprimefaces. [Accessed: 28-Oct-2015].

[24] “Spring Framework.” [Online]. Available: http://projects.spring.io/spring-framework/. [Accessed: 02-Mar-2015].

[25] “JPA - Java Persistence API.” [Online]. Available: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-. [Accessed: 30-Apr-2015].

 ISBN: 978-1-61208-438-1

Intersection of MPS.BR-E and SPICE Models Focused on Projects for the Automotive Industry

Vanessa Matias Leite, Jandira Guenka Palma, Emmanuel da C. Gallo
Department of Computer
Londrina State University
Londrina-PR
Email: vanessa.matiasleite@gmail.com, jgpalma@uel.br, emmanuelcgalo@hotmail.com

***Abstract*—The quality of software is an area of software engineering, which aims to ensure a satisfactory performance in the development of the final product through processes. With the need for better structured processes, models were instituted to steer organizations in their processes, however the adopted models may differ from company to company, in some cases companies become partners, or are suppliers of other, or no merger of companies, so it is important to understand the relationship between the models. This paper presents the intersection of MPS.BR (*Melhoria de Processo do Software Brasileiro*- Brazilian Software Process Improvement) and Automotive SPICE (Software Process Improvement and Capability dEtermination), in addition it relates the processes of a company that adheres to the Brazilian quality model, with automotive reference model. This paper provides a solution for two companies, with different reference models, to work together.**

Keywords- ***Automotive SPICE; MPS.BR; Software Quality.***

I. INTRODUCTION

With the software market on the rise, it was necessary that fashion models processes were established to provide products with higher quality. Today there are various types of process models and a company must analyze and choose a model that will help it increase the quality of its products through changes in their processes.

Process models are not just about the certifications. Their benefits go far beyond that, the institution of international production standards allows higher visibility and gains improving productivity of the company, as it reduces the time and investment used in the projects. However it is important to take note that these results come only upon the maturity of the new companies with the established processes.

Many Original Equipment Manufacturers (OEMs) in the automotive sector has factories in several countries in the world, and many of them have adopted the Automotive SPICE for the development of their products. These industries can hire local suppliers to help them, but need to keep the same instituted quality standards.

In Brazil there is a reference model called MPS.BR. The MPS.BR has seven maturity levels [1]. This model is adopted mainly by small and medium enterprises, and these are even beginning to institutionalize quality processes, and usually are lower maturity levels G, F and E. The company's case study was implementing the level and therefore this work was compared to the level E.

This work will address two types of process quality models, namely the MPS.BR and Automotive SPICE, which is a branch of ISO / IEC 15504 (SPICE). The MPS.BR is a Brazilian model which has as its main objective, the process improvement, being aimed primarily at small and medium-sized businesses. SPICE is an international model itself and its main differential is the focus on improving the design process, not the company as a whole. Many automotive companies use SPICE and a specific aspect of this model was created for these companies, called SPICE Automotive [2].

Given this scenario, the objective is to carry out a study of the compatibility between models of software processes MPS.BR level E and Automotive SPICE acting as a facilitator for companies to see the potential in the Brazilian quality model, and to also show the potential of the Brazilian company to provide services to the automotive industry.

In Section 2 Theoretical Foundations will introduce all the theoretical basis for understanding the work that will be presented. Section 3 the intersection of Automotive SPICE models with MPS.BR and the company will be addressed. Results of analysis are exposed in Section 4. In Section 5 will focus on the approached conclusion of all work.

II. THEORETICAL FOUNDATIONS

Software quality is fundamental for both customers and suppliers, and process models, such as the MPS.BR and SPICE act as guides that assist in improving the software production processes, and thus a means of increasing the quality of the final product.

With the increasing need for more elaborate software, quality becomes an essential item. Pressman defines software quality as conformance to functional requirements and performance that have been explicitly declared, the development patterns clearly documented, and the implicit features that are expected of all software developed by professionals [1]. According to Fuggetta, to guarantee the quality of a software standardization of procedures should be proposed, namely to establish processes that can be defined as a coherent set of policies, organizational structures, technologies, procedures and artifacts needed to devise, develop, deploy and maintain a software product [3].

Due to the increasing demand for software with higher quality, there is a need for more efficient processes. Pressman defines processes as being the Foundation of software engineering allowing the rational and timely software development. He also claims that the software processes are the

 ISBN: 978-1-61208-438-1

basis for the managerial control of software projects and establishes the content in which the technical methods are applied, the work products (documents, data models, reports, forms, etc.) are produced, the milestones are established, the quality is ensured and the modifications are properly managed [1]. Another definition of process is given by Wilson de Paula Filho, that describes a process as something that can be defined with more or less detail and that its stages may have a partial ordering, thus allowing the parallelism between them [4]. As an organization matures, its software processes become more defined and consistent across the organization as a whole [5]. The quality of software is related only to a quality of the product, not making connections the way it was produced, or if there is a process that assisted in the manufacture of the same. On the other hand, when one looks at the quality of a software process, we seek to identify the ability of a software process to produce a quality product [6].

However, it is important to notice that the mere existence of a software process does not guarantee that a quality product will be created. For the results to be achieved in a satisfactory manner, it is necessary that the processes are well defined and understood. The process adopted must be used in all projects of the Organization, being appropriate to the particular characteristics of each project [7].

The software process model is an abstract representation of the architecture, design or definition of the software processes [8]. This work focuses on two models of software processes, the MPS.BR and Automotive SPICE. These models provide a set of core activities to obtain a software within what was proposed, however they do not specify a possible life cycle or how to put into practice such activities, so it is up to the Organization to model the processes within your reality[7][9].

Software process models are usually divided into maturity levels, which can be defined as specific or generic practices related to a pre-defined set of processes that improve the overall performance of an organization [10].

From these definitions, it is possible to conclude that the software processes are fundamental in an organization and the implementation of the same provides benefits in various sectors, mainly in the quality of the developed software. However, the implementation of processes without well-defined methods is usually a chore, this leads to process models, which provide a set of policies, organizational structures, procedures, among other elements that assist in this task.

A. MPS.BR

The Brazilian Software Process Improvement also called MPS.BR, is an evaluation model of software development companies developed by SOFTEX in 2003 [11], in partnership with the Brazilian federal government and the academic community. The Brazilian model is independent, but compatible with standards ISO 12207 and 15504, CMMI (North American model of process improvement) [12].

The MPS.BR is of great importance for the Brazilian scenario, since 94% of Brazilian companies are considered small and medium businesses [13]. Therefore, for these organizations a certification of an international model becomes costly and often impractical. The Brazilian quality model provides an effective way to process definition and certification costs affordable to the local reality. The MPS.BR is divided into seven levels of maturity. The maturity levels are: A, B, C, D, E, F, G. Level A is the highest level of the reference model. Each level of maturity has processes or activities that must be performed. The processes contained until the level E are reported in the Table I. With each new level of maturity, process or improvement processes that have already been established in previous levels are added. One advantage of the Brazilian model being divided into several levels is the deployment of more gradual fashion model, favoring its adoption by small and medium-sized enterprises [14].

TABLE I. PROCESSES OF THE MPS.BR-E

Level	Name	Processes
G	Partially Managed	Requirements management – GRE Project management –GPR
F	Managed	Measurement - MED Quality assurance –GQA Configuration management - GCO Acquisition -AQU Project Portfolio Management - GPP
E	Partially Defined	Evaluation and Improvement of the Organizational Process - AMP Organizational process definition - DFP Human resources management - GRH Reuse management – GRU

B. SPICE

The SPICE or ISO/IEC 15504 was originally created as a supplement to the ISO/IEC 12207, and it aims to guide the assessment and self-evaluation of the ability of companies in the processes and from this evaluation the improvement of its processes is enabled [15]. This model of software quality establishes a framework which is used for both the creation of evaluation processes and the improvement of software processes [16].

The ISO/IEC 15504 is structured in two dimensions. The first is the dimension of processes in which the processes are evaluated. The second is the capacity dimension that determines the ability of the company to evaluate in each of these processes [15].

The company does not need to execute all processes; the organization can opt for a subset of processes that match their needs. Thus, in an assessment, the processes can have different levels [17].

The five major categories within the dimension of SPICE processes are [15]:

- CUS: Customer/vendor relationship
- ENG: Engineering processes
- SUP: Support processes
- MAN: Management processes
- ORG: Organization processes

The SPICE capacity dimension possesses 6 levels [16]:

- 0- Incomplete
- 1- Process performed
- 2- Managed process
- 3- Process established

 ISBN: 978-1-61208-438-1

- 4- Predictable Process
- 5- Optimized Process

C. Automotive SPICE

Currently it is estimated that about 85% of the functionality of an automobile are controlled by software. Auto companies need these softwares more reliable thus avoiding future problems like Recalls. For this reason, from a consensus among the major automakers, the Automotive SPICE was created, aiming to assist in the production of automotive software and based on the ISO/IEC 15504 (SPICE) [18]. In addition to the procedures defined by ISO/IEC 15504, the Automotive SPICE defines alterations in the ISO processes and inserts other suitable processes and a focus on the needs of the automotive industry [19].

The groups of processes that the Automotive SPICE treats and will be addressed in this paper are in Tables II-VIII. Each process contains process outcomes and base practices to assist in understanding and implementing processes.

TABLE II. ACQUISITION PROCESS GROUP (ACQ)

Acronym	Name
ACQ.3	Contract agreemen
ACQ.4	Supplier monitoring
ACQ.11	Technical requirements
ACQ.12	Legal and administrative requirements
ACQ.13	Project requirements
ACQ.14	Request for proposals
ACQ.15	Supplier qualification

TABLE III. SUPPLY PROCESS GROUP (SPL)

Acronym	Name
SPL.1	Supplier tendering
SPL.2	Product release

TABLE IV. ENGINEERING PROCESS GROUP (ENG)

	Nome
ENG.1	Requirement elicitation
ENG.2	System requirements analysis
ENG.3	System architectural design
ENG.4	Software requirements analysis
ENG.5	Software design
ENG.6	Software construction
ENG.7	Software integration test
ENG.8	Software testing
ENG.9	System integration test
ENG.10	System testing

TABLE V. SUPPORTING PROCESS GROUP (SUP)

Acronym	Name
SUP.1	Quality assurance
SUP.2	Verification
SUP.4	Joint review
SUP.7	Documentation management
SUP.8	Configuration management
SUP.9	Problem resolution management
SUP.10	Change request management

TABLE VI. MANAGEMENT PROCESS GROUP (MAN)

Acronym	Name
MAN.3	Project management
MAN.5	Risk management
MAN.6	Measurement

TABLE VII. PROCESS IMPROVEMENT PROCESS GROUP (PIM)

Acronym	Name
PIM.3	Process improvement

TABLE VIII. REUSE PROCESS GROUP (REU)

Acronym	Name
REU.2	Reuse program management

D. The company

The company which processes were studied is a small company located in the city of Londrina-PR, It works with firmware development, test and validation to automotive industry. With ten years of activity, this organization possesses certification MPS.BR-F and comes pleading the level E of the Brazilian model. In addition to these achievements, the company also has MoProSoft certification, a Mexican model of quality process [20].

III. LIST OF AUTOMOTIVE SPICE PROCESS WITH MPS.BR-E AND WITH PROCESS COMPANY

In this section the relationship between the processes of Automotive SPICE with the MPS.BR level E it will be addressed, as well as the comparison with the already established processes within the studied company. The correlation between the two models of references was developed considering each Automotive SPICE process and checking the corresponding in MPS.BR. The same analysis was performed with the company's processes, but only if the company complied fully, partially or did not fulfill the requirements presented in the templates.

This intersection was illustrated below through the Automotive SPICE's PIM.3 process and its compatibility with the MPS.BR's processes, and finally, the compatibility of the PIM.3 with the company's processes.

 ISBN: 978-1-61208-438-1

PIM.3 - Process Improvement

Process Purpose [18]: "The purpose of the Process improvement process is to continually improve the organization's effectiveness and efficiency through the processes used and aligned with the business need."

Process Outcomes

As a result of successful implementation of this process:

1) commitment is established to provide resources to sustain improvement actions;

2) issues arising from the organization's internal/external environment are identified as improvement opportunities and justified as reasons for change;

3) analysis of the current status of the existing process is performed, focusing on those processes from which improvement stimuli arise;

4) improvement goals are identified and prioritized, and consequent changes to the process are defined, planned and implemented;

5) the effects of process implementation are monitored, measured and confirmed against the defined improvement goals;

6) knowledge gained from the improvement is communicated within the organization; and

7) the improvements made are evaluated and consideration given for using the solution elsewhere within the organisation.

The compatibility of MPS.BR with each item listed above is presented in the following:

1) It is not addressed in MPS.BR level E;

2) The identification of the organization's issues are treated in AMP1;

3) The analysis of the current state is covered in AMP3;

4) The identification of improvements and prioritization are treated in AMP5, already planning the implementation is contained in AMP6;

5) The effect of improvement implemented is specified AMP6;

6) It is treated with relevance in MPS.BR E;

7) Evaluations and considerations are discussed in AMP9.

Compatibility with the Company:

1) It is not implemented by the company;
2) The company has a process called Definition and Process Improvement, in which executes this request.
3) The company has a process called Definition and Process Improvement, in which executes this request.
4) The company has a process called Definition and Process Improvement, in which executes this request.
5) The company has a process called Definition and Process Improvement, in which executes this request.
6) The company has a process called Definition and Process Improvement, in which executes this request.
7) It is not implemented by the company;

All processes of the Automotive SPICE were related in this same manner in Leite [21], as with MPS.BR as with the company.

These processes are summarized in the Tables IX-XV and were created comparing the definitions employed by standards of the Automotive SPICE [18] with the MPS.BR E defined by SOFTEX guides [11], and with the processes of the organization. The Tables relate the processes of Automotive SPICE, the processes of the MPS.BR-E and if there are processes in the company, being that this relationship can be addressed in three ways: completely, partially or there may not be processes in MPS.BR-E used by the company that meets what is determined Automotive SPICE

The following Tables with the intersection of processes.

TABLE IX. INTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR ACQ.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
ACQ.3	Yes	AQU3, AQU4, AQU6	No
ACQ.4	Yes	AQU4, AQU6	No
ACQ.11	No		No
ACQ.12	Partially	AQU4	No
ACQ.13	No		No
ACQ.14	No		No
ACQ.15	No		No

TABLE X. INTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR ENG.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
ENG.1	Partially	GRE1, GRE4, GRE5	Partially
ENG.2	Partially	GRE3, GER4, GRE5	Partially
ENG.3	No		Yes
ENG.4	No		No
ENG.5	No		Partially
ENG.6	No		No
ENG.7	No		No
ENG.8	No		No
ENG.9	No		No
ENG.10	No		No

 ISBN: 978-1-61208-438-1

TABLE XI. INTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR SUP.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
SUP.1	Partially	GQA1, GQA2, GQA3, GQA4	Yes
SUP.2	No		Yes
SUP.4	No		Yes
SUP.7	No		Partially
SUP.8	Yes	GCO1, GCO2, GCO3, GCO5, GCO6, GCO7	Yes
SUP.9	Partially	GPR18, GPR19	Yes
SUP.10	Partially	GCO5	Partially

TABLE XII. INTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR SPL.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
SPL.1	Partially	GPR1, GPR2, GPR4, GPR5, GPR6, GPR7, GPR8, GPR9, GPR11, GPR12	Yes
SPL.2	Partially	GCO1, GCO2, GCO6, GCO7	Yes

TABLE XIII. NTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR MAN.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
MAN.3	Partially	GPR1, GPR2, GPR4, GPR8, GPR13, GPR19	Yes
MAN.5	No		Partially
MAN.6	Yes	MED1, MED2, MED 5, MED6, MED7	Partially

TABLE XIV. INTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR PIM.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
PIM.3	Partially	AMP1, AMP3, AMP5, AMP6, AMP9	Partially

TABLE XV. INTERSECTION OF MODELS AUTOMOTIVE SPICE , MPS.BR-E AND ORGANIZATION PROCESSES FOR REU.

Automotive SPICE Process	Approached by MPS.BR-E?	MPS.BR-E	Company
REU.2	Partially	GRU1	Partially

IV. ANALYSIS OF RESULTS

With the basis of the information presented in the Tables IX to XV, one can extract some considerations. In Figure 1, the result of the compatibility between the processes of the MPS.BR are exposed and adopted by the Organization in relation to the Automotive SPICE.

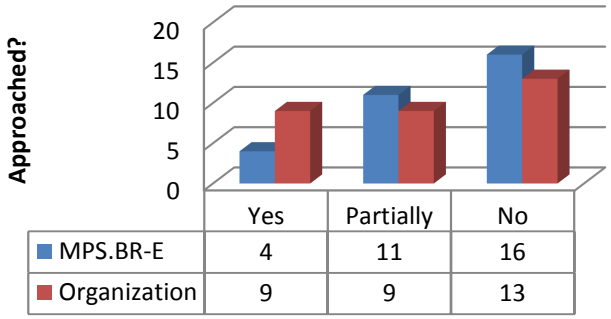


Figure 1. Processes Compability Results

These results show that the Brazilian quality model in the level E was not as compatible with the automotive model. However, it is noticed that the existing processes within the company have a more significant compatibility.

The above report exposes a remarkable fact, the organization, which should theoretically have a similar result with the quality of Brazilian model, because it has implemented the level E, got more processes completely addressed in relation to the Automotive SPICE, it shows that even with the certification of a quality model, the company has to fit and supply what is needed and what is not exposed in the implementation of a quality model guides

The factor that may have influenced the company to greater compatibility with the SPICE is in fact the organization provides services to the automotive industry for almost 10 years constantly, dealing with the high level of requirements of certain customers. In addition, the company also has another certification of quality model that is the MoProSoft.

Another analysis that can be extracted from this compatibility is that the level E from MPS.BR has 11 processes, of which 8 were used at the intersection with the Automotive SPICE. Based on this opinion, it can be concluded that the majority of Brazilian model processes have been addressed, so the level E from MPS.BR discussed in this work brings a few contributions when compared to automotive model.

V. CONCLUSION

This paper showed that quality is an important element when it comes to software and processes tend to bolster its development, granting in this way an accuracy and standardization by assigning a higher quality to the final product. Software process models provide guidance and precepts that assist in the preparation and even implementing processes.

With the analysis of the intersection between models and company, it can be concluded that among the Automotive SPICE and the MPS.BR-E the compatibility was little expressive, when looking at the number of processes that were addressed in the Brazilian model, it was concluded that the level covered in this work is insufficient on automotive model processes. Probably there will be greater compatibility when compared with the higher maturity levels the MPS.BR.

The organization compatibility was more significant. Domestic needs, customer requirements, certification of other

 ISBN: 978-1-61208-438-1

quality model, providing services for the automotive area are some of the factors that influence the company's processes, it resulted in a more complete processes and therefore in greater compatibility.

Through the analysis a question arises. Should the company change its processes to meet the Automotive SPICE? As it was exposed, the necessary changes to an organization that adopts the MPS.BR-E would be rough, the company must analyze its services, if the majority of its projects do not have the focus to the automotive sector, the organization may choose to sell this differential when needed, not changing its processes.

The big companies, or international certifications, will not always be found in environments where industries are located, so how to seek suppliers? It is concluded that suppliers with certifications will be better able to provide services than those without, as well as to assimilate/attend processes arising from the contractor.

With the compatibility of quality models and the company made throughout this work, it obtained a document that could assist other organizations that might adopt the MPS.BR, and become a facilitator so that a company can see its potential in front of a model of international quality and provide services to the automotive sector. Another factor that it is necessary to emphasize, is that a company with this intersection relationship has the possibility to analyze the processes that are not addressed by the Brazilian model and adapt to the automotive model without necessarily aiming at a certification, and thus a differential in the market. Thus, it was concluded that the work reached the objectives proposed, since it was held the intersections of Automotive SPICE process with the MPS.BR-E and with the process of an organization.

ACKNOWLEDGMENT

This work was supported by CNPQ (National Council for Scientific and Technological Development -*Conselho Nacional de Desenvolvimento Científico e Tecnológico*), in the project ITI (Technological industrial Initiation) together with other project productivity scholarship.

REFERENCES.

[1] S. Roger, Software Engineering (Engenharia de Software), 6a edition. McGraw-Hill, 2006.

[2] M. Griesser, F. Schreiner, and S. Stölzl, “Applying Functional Safety Management and SPICE for Automotive Functions”, Safety, vol. 2012, 2008, pp.03–21.

[3] A. Fuggetta, “Software Process: a Roadmap”, in Proceedings of the Conference on the Future of Software Engineering.ACM2000, pp.25-34.

[4] W. d. P. Paulo Filho, Software Engineering: fundamentals, methods and techniques (Engenharia de Software: fundamentos, métodos e técnicas). Rio de Janeiro: LTC, 2003.

[5] M. Paulk, Capability Maturity Model for Software. Wiley Online Library, 1993.

[6] G. A. García-Mireles, M. Ángeles Moraga, F. García, and M. Piattini, “approaches to promote product quality within software process improvement initiatives: A mapping study”, Journal of Systems and Software, 2015.

[7] S. Weber, J. C. R. Hauck, and C. V. Wangenheim, “Establishing software processes are micro and small enterprises” (“Estabelecendo processos de software em micro e pequenas empresas”), SBQS-Simpósio Brasileiro de Qualidade de Software, Porto Alegre, Brazil, 2005.

[8] P. H. Feiler and W. S. Humphrey, “Software process developmentand enactment: Concepts and definitions”, in Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the. IEEE, 1993, pp. 28–40.

[9] S. T. Acuña and X. Ferré, “Software process modelling”. in ISAS-SCI (1), 2001, pp. 237–242.

[10] F. J. Pino, M. T. Baldassarre, M. Piattini, and G. Visaggio, “Harmonizingmaturity levels from cmmi-dev and iso/iec 15504”, Journal of Software Maintenance and Evolution: Research and Practice, vol. 22, no. 4, 2010, pp. 279–296.

[11] SOFTEX, “General guide software” (“Guia Geral de Software”), 2012. [Online]. Available: http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf [retrieved: 09, 2015].

[12] Montoni, Mariano, et al. “Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR”, in Product-Focused Software Process Improvement. Springer, 2006, pp. 249–262.

[13] ABES – Associação Brasileiro de Empresas de Software, “Brazilian market of panorama software and trends” (“Mercado brasileiro de software panorama e tendências”), 2006.

[14] K. C. Weber, E. Araújo, C. Machado, D. Scalet, C. F. Salviano, and A. R. C. d. Rocha, “Reference model and evaluation method for improving software-version 1.0 process (MR-MPS and MA-MPS)” (“Modelo de referência e método de avaliação para melhoria de processo de software-versão 1.0 (MR-MPS e MA-MPS)”), IV Simpósio Brasileiro de Qualidade de Software. Porto Alegre-RS: Anais do SBQS, vol. 2005, 2005, p. 14.

[15] R. S. Wazlawick, Software Engineering: concepts and practices (Engenharia de Software: conceitos e práticas). Rio de Janeiro: Elsevier, 2013.

[16] A. M. L. de Vasconcelos, A. C. Rouiller, C. Â . F. Machado, and T. M. M. de Medeiros, “Introduction to software engineering and software quality” (“Introdução à engenharia de software e à qualidade de software”), 2006.

[17] K. E. Emam and A. Birk, “Validating the ISO/IEC 15504 measures of software development process capability”, Journal of Systems and Software, 2000.

[18] A. SIG, “Automotive SPICE, process reference model,” Jun. 2010. [Online]. Available: http://www.automotivespice.com/fileadmin/software-download/automotiveSIG_PAM_v25.pdf [retrieved: 09, 2015].

[19] M. Mueller, K. Hoermann, L. Dittmann, and J. Zimmer, Automotive SPICE in practice: surviving implementation and assessment. O’Reilly Media, Inc, 2008.

[20] H. Oktaba, "3.2 MoProSoft®: A Software Process Model for Small Enterprises." International Research Workshop for Process Improvement in Small Settings. 2006.

[21] V. Leite, “Intersection of MPS.BR-E and SPICE models focused on projects for the automotive industry " (“Intersecção dos Modelos MPS.BR-E e SPICE com foco em projetos para indústrias do setor automotivo”).Monograph, 2014.

 ISBN: 978-1-61208-438-1

Quality-Based Score-level Fusion for Secure and Robust Multimodal Biometrics-based Authentication on Consumer Mobile Devices

Mikhail Gofman, Sinjini Mitra, Kevin Cheng, and Nicholas Smith
California State University, Fullerton, California, USA
Email: {mgofman, smitra}@fullerton.edu and {kevincheng99, nicholastoddsmith}@csu.fullerton.edu

***Abstract*—Biometric authentication is a promising approach to access control in consumer mobile devices. Most current mobile biometric authentication techniques, however, authenticate people based on a single biometric modality (e.g., iPhone 6 uses only fingerprints), which limits resistance to trait spoofing attacks and ability to accurately identify users under uncontrolled conditions in which mobile devices operate. These challenges can be alleviated by *multimodal biometrics* or authentication based on multiple modalities. Therefore, we develop a proof-of-concept mobile biometric system which integrates information from face and voice using a novel score-level fusion scheme driven by the quality of the captured biometric samples. We implement our scheme on the Samsung Galaxy S5 smartphone. Preliminary evaluation shows that the approach increases accuracy by 4.14% and 7.86% compared to using face and voice recognition individually, respectively.**

Keywords–Multimodal biometrics; quality; score-level fusion; mobile

I. INTRODUCTION

Biometric authentication is the science of identifying people based on their physical and behavioral traits, such as face and voice. Recent advances in mobile technology have enabled such authentication in consumer mobile devices. Although generally regarded as more secure than passwords, most state-of-the-art mobile biometric authentication approaches are unimodal: they identify people based on a single trait. To bypass a unimodal system, an attacker only needs fabricate the single trait the system uses for identification [1]. Unimodal mobile biometric systems also have difficulty accurately recognizing users in conditions known to distort the quality of biometric images (e.g., poor lighting affecting the visibility of a face [2]).

We present the design, implementation, and performance evaluation of a proof-of-concept system to demonstrate that a promising approach to improve the security and robustness of mobile biometric authentication is *multimodal biometrics*, which uses multiple traits to identify people. Our contributions are as follows:

1) We study the effects of face and voice sample quality on recognition accuracy in mobile devices.
2) We develop a multimodal biometric system integrating information from face and voice through a novel *quality-based score-level fusion* scheme, which improves recognition accuracy by letting the modality with higher quality sample have a greater impact on the authentication outcome. Such a scheme lets the system adapt to varying background conditions, which affect the quality of biometric images.
3) We evaluate the system using our database of face and voice samples captured using a Galaxy S5 smartphone in a variety of background conditions. The results indicate that the approach achieves higher recognition accuracy than unimodal approaches based solely on face or voice.

The rest of the paper is organized as follows. Section II discusses the role of quality in mobile biometric authentication, Section III introduces multimodal biometric systems. We present our quality-based score-level fusion scheme in Section IV followed by the results in Section V. Finally, we conclude in Section VI.

II. ROLE OF QUALITY IN MOBILE BIOMETRICS

A low-quality biometric sample, such as low resolution face photograph or noisy voice recording, can cause a biometric algorithm to incorrectly identify an impostor as a legitimate user (*false acceptance*) or a legitimate user as an impostor (*false rejection*). Capturing high-quality samples on mobile devices is especially difficult because (i) people often operate mobile devices in insufficiently lit and noisy environments (e.g., malls and restaurants), choose less-than-optimal camera angles, and might have dirty fingers [2]; and (ii) biometric sensors in consumer mobile devices often trade sample quality for portability and lower costs, leaving them vulnerable to trait spoofing attacks [3]. We believe that these challenges can be addressed via multimodal biometrics.

III. MULTIMODAL BIOMETRICS ON MOBILE DEVICES

Multimodal biometrics require users to authenticate using multiple relatively-independent traits, which adds layers of security by forcing attackers to fabricate multiple traits. Also, identifying information from modalities with high-quality images can compensate for the missing/inaccurate identifying information in low-quality images from other modalities.

In multimodal biometric systems, information from different modalities can be consolidated (i.e., fused) at the decision-, match score-, or feature-level [6]. We integrate face and voice modalities using score-level fusion, because it is considered more effective than decision-level fusion and less complex and computationally expensive than feature-level fusion.

IV. QUALITY-BASED SCORE-LEVEL FUSION

Sample quality can drastically affect recognition accuracy; therefore we integrate it into our multimodal scheme. We assess the quality of face images based on luminosity, sharpness, and contrast [4] and use the signal-to-noise ratio (SNR) approach [5] to assess the quality of voice samples. Once assessed, the metrics are normalized using the z-score normalization method. This particular method was selected since it is a commonly used normalization method, is easy to implement, and is highly efficient [9].

For face recognition, we use *FisherFaces*, which works well when images are captured under varying conditions, as is the case with mobile devices [7]. The algorithm uses pixel intensities in the image as identifying features. For voice recognition, we use Mel-Frequency Cepstral Coefficients (MFCCs) as the identifying features in a Hidden Markov Model (HMM)-based identification method [8]. After training

 ISBN: 978-1-61208-438-1

these algorithms with samples from users, they are used to match samples supplied during authentication, and are the basis of the score-level fusion scheme, as described below.

Let t_1 and t_2 denote the quality scores for the face and voice samples in the training data, respectively. During authentication, we calculate the quality scores Q_1 and Q_2 of the two biometrics from the test data and determine their proximity to t_1 and t_2, respectively. We then compute the weights of the face and voice modalities w_1 and w_2, as $w_i = \frac{p_i}{p_1+p_2}$, so that w_1 + w_2 = 1, where p_1 and p_2 are percent proximities of Q_1 to t_1 and Q_2 to t_2, respectively. The closer Q_i is to t_i, the greater is the weight assigned to the corresponding modality, which ensures the effective integration of quality in the final authentication process. Next, the matching scores S_1 and S_2 are obtained from face and voice recognition algorithms. The overall match score is then computed using the *weighted sum rule*: $M = S_1w_1 + S_2w_2$. If $M \geq T$ (T: pre-selected threshold), the system accepts the person as authentic; otherwise, it declares the person an imposter.

While using the above scheme, it is important to exercise caution to ensure significant representation of both modalities in the fusion process. For example, if Q_2 differs greatly from t_2 but Q_1 is close to t_1, the face modality will dominate the authentication process, resulting in a nearly unimodal scheme based on the face biometric. Thus, a mandated benchmark is required for each quality score to ensure that the system denies access if the benchmarks for both scores are not met.

V. Results

A. The Dataset

Due to the unavailability of a diverse multimodal mobile biometric database, we created one with videos from 54 people of different genders and ethnicities. They held a phone camera in front of their face while saying a certain phrase. The videos were recorded in various real-world settings using a Samsung Galaxy S5 smartphone. The faces display the following variations: (1) four expressions: neutral, happy, sad, angry, and scared; (2) three poses: front and sideways (left and right); and (3) two illumination conditions: uniform and partial shadows. The voices in videos have different levels of background noise, from traffic noises to music and chatter, and voice distortions like raspiness. Twenty popular phrases were used (e.g., *unlock* and *football*). The database is still in development and will be made available to researchers upon completion.

B. Performance Results

We implemented our score-level fusion scheme on the Android-based Samsung Galaxy S5 device. Table I shows preliminary performance results. We measure recognition accuracy using equal error rate (EER), which is traditionally used in biometrics applications and is the value that produces the best possible combination of the False Acceptance Rate (FAR) and False Rejection Rate (FRR) (i.e., where FAR and FRR are equal) [6]. The final results were obtained by selecting a random set of five users from the database and training the face and voice algorithms with 40 face images and 40 voice samples of these users. Most samples were automatically extracted from one good-quality video and few samples were extracted from low-quality videos. For testing, we used 80 combinations of randomly selected face frames and voice samples from videos of varying quality. The experiment was repeated for 1000 different training/testing combinations of users, and the face, voice, and score-level fusion EERs and training and authentication execution times were average. According to the table, our quality-based score-level fusion approach improves accuracy by 4.14% and 7.86% compared unimodal face and voice recognition approaches, respectively.

Note: Good-quality training samples are important because their quality metrics provide a baseline for judging qualities of samples supplied during authentication. Adding a few noisy training samples also increases the chances of recognizing the user in similar noisy conditions. Also, although the training times are longer than authentication times (common in classification problems), training happens only once when the user registers his/her training data with the device. Authentication is real-time and should require less time, as is the case here.

TABLE I. QUALITY-BASED SCORE-LEVEL FUSION EER RESULTS.

Modality	EER	Training Time (sec)	Auth. Time (sec)
Face	18.70%	575.491	0.2133
Voice	22.42%	295.692	0.0728
Score-level Fusion	14.56%	871.183	0.2861

C. Analysis of Sample Quality

We briefly discuss the quality of face and voice samples in our database and its complex relationship to recognition accuracy. We find that face luminosity and contrast metrics exhibit bimodal distributions caused by different conditions, such as shadows. Voice SNR exhibits normal distribution for good-quality samples, sick voices, and voices with chatter in the background. These distributions can provide useful guidance for designing automatic sample quality enhancement mechanisms in mobile devices. We also observe that variations in luminosity and pose are greater challenges to minimizing the face recognition EER, compared to sharpness and contrast. However, there are important exceptions such as, images distorted by motion blur, matching poorly due to the differences in sharpness despite similar luminosity. These findings illustrate the complex sample-quality challenges facing mobile biometrics. They also lay the groundwork for devising a statistical framework for predicting optimal modality weights in our scheme and determining the quality thresholds for acceptable error rates.

VI. Conclusions and Future Work

The preliminary results show that multimodal biometrics can improve biometrics-based authentication in consumer mobile devices. Our next step is to refine the method to reduce EER more and incorporate other modalities (e.g., ears and fingerprints).

References

[1] D. Smith, A. Wiliem, and B. Lovell, "Face recognition on consumer devices: Reflections on replay attacks," IEEE Transactions on Information Forensics and Security, vol. 10, 2015, pp. 736–745.

[2] C. Bhagavatula, B. Ur, K. Iacovino, S. M. Kywe, L. F. Cranor, and M. Savvides, "Biometric authentication on iPhone and Android: Usability, perceptions, and influences on adoption," presented at Proceedings of the NDSS Workshop on Usable Security 2015 (USEC). San Diego, California, February 2015. URL: http://www.blaseur.com/papers/usec15_talk.pdf [accessed: 2015-25-8].

[3] "The trouble with Apples Touch ID fingerprint reader, Wired.com, December 3, 2013, URL: http://www.wired.com/2013/12/touch-id-issues-and-fixes/ [accessed: 2015-25-8].

 ISBN: 978-1-61208-438-1

[4] K. Nasrollahi and T. B. Moeslund, “Face Quality Assessment System in Video Sequences,” Biometrics and Identity Management: First European Workshop (BIOID), May 7-9, 2008 Roskilde, Denmark. Springer Berlin Heidelberg, 2008, pp. 10–18, URL: http://dx.doi.org/10.1007/978-3-540-89991-4_2 [accessed: 8-25-2015].

[5] M. Vondrášek and P. Pollak, “Methods for speech SNR estimation: Evaluation tool and analysis of VAD dependency,” Radioengineering, vol. 14, no. 1, 2005, pp. 6–11.

[6] A. K. Jain and A. Ross, “Multibiometric systems,” Communications of the ACM, vol. 47, no. 1, 2004, pp. 34–40.

[7] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, 1997, pp. 711–720.

[8] D. Hsu, S. M. Kakade, and T. Zhang, “A spectral algorithm for learning Hidden Markov Models,” Journal of Computer and System Sciences, vol. 78, no. 5, 2012, pp. 1460–1480.

[9] A.K. Jain, K. Nandakumar, and A. Ross, “Score normalization in multimodal biometric systems,” *Pattern Recognition*, 2005, Vol. 38, pp. 2270–2285.

 ISBN: 978-1-61208-438-1

An Approach for Sensor Placement to Achieve Complete Coverage and Connectivity in Sensor Networks

Monia Techini

University of Gabes,
National Engineering School
of Gabes (ENIG),
Av. Omar Ibn El Khattab
Zerig-6029, Gabes, Tunisia
Email: moniatec@gmail.com

Ridha Ejbali and Mourad Zaied

REGIM-Lab: REsearch Groups
in Intelligent Machines
BP 1173, Sfax, 3038, Tunisia
Email: ridha_ejbali@ieee.org

Email: mourad.zaied@ieee.org

***Abstract*—The main objective required for a Wireless Sensor Network (WSN) is the positioning of the sensor nodes to form a WSN. There are two strategies for the positioning of nodes in an area of interest where some phenomena are to be monitored: deterministically or randomly. In this paper, our goal is to propose a new sensor placement technique, which is based on random distribution with coverage and connectivity constraints.**

Keywords–Placement; Random; WSN; Coverage; Connectivity.

I. INTRODUCTION

Recent developments could not be achieved without a change in the field of communication. Mainly Wireless Sensor Networks (WSN) and mobile computing become more and more popular. Control of environmental parameters can give rise to several applications. In the monitoring area, the WSN is deployed over a region where some phenomena have to be monitored. The nodes can be equipped with sensors to measure temperature, humidity, and gases.

A WSN consists of a number, often significant, of sensors with limited perception and communication capabilities where each node is connected to one (or sometimes several) sensors. There exist two types of sensors: homogeneous sensors, which possess the same communication and computation capabilities and heterogeneous sensors, which possess different capabilities. Determining the sensor field architecture is a key challenge in sensor resource management. Sensors have to be placed at critical locations that provide sustainable coverage.

Many works for sensors and sensor placement were done. In [1], Lin models the sensing field as a grid points. Then, he explains how to place sensors on some grid points in order to satisfy a particular quality of service based on the Simulated Annealing approach. Using the genetic algorithm, the authors in [2] propose an approach for the sensor placement problem. In [3], the Improved Particle Swarm Optimization (IPSO) algorithms have been employed to determine the optimal sensors number and configurations.

Recent works focused on relay placement algorithm to determine the set of positions which can guarantee connectivity. The main objective of the work proposed in [4] is to reduce the set of locations for the existing mobile nodes in the network to the locations of relay nodes that would ensure connectivity with the least count. A multi-objective mathematical model is used to determine the best placement of mobile nodes for different tasks [5]. In [6], Flushing proposes a combination of exact method and heuristic method to solve the problem mentioned below. Thus, his work was based on Mixed-Integer Linear Programming (MILP) and a Genetic Algorithm (GA).

A related problem to the deployment in WSN is Art Gallery Problem (AGP) addressed by the art gallery theorem [7]. The AGP problem consists of determining the minimum number of guards to cover the interior of an art gallery. Many researches for the AGP have been discussed in literature. There exist some similarities between our sensor placement problem and the AGP, such that the guards (sensors in our case) are assumed to have similar capabilities.

As we mentioned earlier, there exist two ways for the placing of sensors with random placement or with grid-based placement. The former one is defined as a technique where nodes are deployed at random positions. The latter is defined as the one where sensors are placed exactly at pre-engineered positions. Figure 1 shows the different categories of node placement strategies.

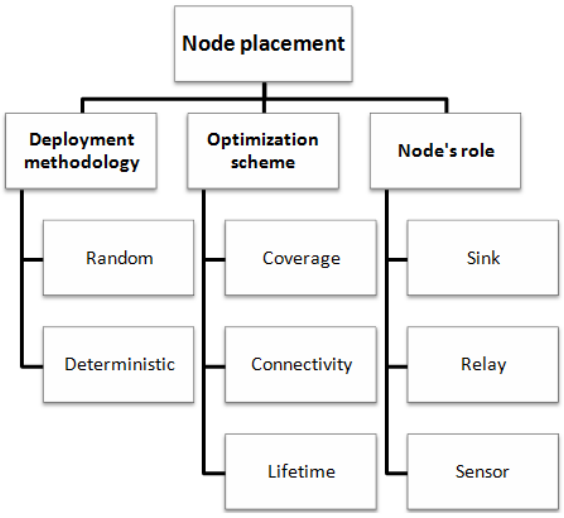


Figure 1. Sensor node placement methodologies

On the other side, the authors in [8] classify the place-

 ISBN: 978-1-61208-438-1

ment strategies into static and dynamic depending on whether the optimization is performed at the time of deployment or while the network is operational, respectively. In [8], Younis assumes that the choice of the deployment scheme depends on many properties. Thus, many researches assume that for some cases, the random placement becomes the only option due to environment characteres [1] and deployment cost and time [9]. This paper focuses on the implementation of an optimal node deployment strategy based on a random placement method. The remaining part of this paper is organized as follows: Section 2 starts by summarising the problem of the sensor placement. Our sensor placement approach is then presented in Section 3. Section 4 shows the results of our procedure. Section 5 provides our conclusions.

II. Problem statement

The problem addressed in this paper has as objective to determine minimum number of sensors when the maximum of targets are covered. There are two strategies of sensor node placement: deterministic or non-deterministic (random). We assume our goal is to deploy the sensors in order to provide the connectivity and the coverage of the service area. Thus, the main objective of our approach is to ensure that the maximum of the area is covered. The authors in [10] formulate a constrained multivariable nonlinear programming problem to deploy the sensors in their locations under constraints of network lifetime and total power consumption. They propose two optimal placement strategies to this problem.

The number of the sensors is determined by taking into account both the coverage of the area of interest and the communication between sensors in the WSN. Thus, communication in a WSN is an effective and practical way to improve the system performance of WSN. This suggests that both the network connectivity and the coverage are an essentiel parameter which must be taken into account when positioning sensors. Each deployed sensor should be able to cover an area in order to increase the coverage in a defined region while maintaining the connectivity in the WSN.

III. Method

Sensor placement can greatly impact the WSN performances, such as coverage and connectivity. The former is defined to quantify the quality of service (QoS). The latter answers the questions about the communication network in a WSN. Table I introduces the parameters used during our work.

TABLE I. Parameters of the WSN.

Parameter	*Definition*
C_i	Node i
R_s	Node sensing radius
R_c	Node communication radius
A	Network area
A_i	Area of the Voronoi polygon for a sensor
N	Number of sensor nodes in the network
C	Coverage of the total area
D_{ij}	Distance between nodes i and j
N_e	Maximim number of neighbors

Minimizing the number of deployed sensors and maintaining higher sensor coverage when positioning the sensors have always been a challenge, especially when the monitoring region is unknown. The aims of the present article are to implement an approach for the sensor placement problem with coverage and communication constraints. The main goal of the sensor placement approach is to determine the best placement of the sensors. In addition, the proposed algorithm aims at achieving high sensor coverage while maintaining network connectivity.

We assume that a region A is covered if there exists a sensor at position p with sensing radius R_s that contains the region A completely [1]. Thus, the ability of detection varies with the distance between the sensor C_i and the target T_j [11] [12], mathematically speaking we have:

$$D_{i,j} \leq R_s \tag{1}$$

where $D_{i,j}$ is the distance between the position of the sensor C_i and the target T_j.

To guarantee node communication, The connectivity is assumed to be full if the distance between two sensors is less than the communication radius of the sensor. The distance is defined as the Euclidean distance between two sensors. Figure 2 shows a pseudo code of the algorithm.

```
Algorithm PLACE_SENSORS:
Step 0: deploy first sensor C_0 randomly
Step1: while a maximum coverage not achieved
Step2: generate a random position for sensor C_i
      Evaluate the distance D_ij between C_i and C_j  (0 ≤ j < i)
      If exist C_j  while D_ij ≤  R_c then
        If C_j 's neighbors ≤ Ne
          Deploy C_i
          Link C_i and C_j and go to Step1
        Else
          Go to Step2
        End
      Else
        Go to Step2
      End
End
```

Figure 2. Algorithm for sensors placement

We begin by generating a random position for the first sensor. Then, deploying the rest by taking into account a maximum coverage and connectivity in the area of interest as constraints. The number of neighbors of each deployed sensor should be less than a defined number N_e in order to ensure a sufficient distribution in the area.

Process node 1: deploys the first sensor randomly.

Process node i: node i is deployed if there is at least a link with node j ($0 \leq j \leq i-1$) and a sufficient coverage not achieved yet.

 ISBN: 978-1-61208-438-1

The algorithm is iterative, and it places one sensor in the sensor field during each iteration. It terminates either when a sufficient coverage of the zone is achieved.

IV. Simulation And Results

The proposed method aims at defining the initial placement of sensors. All sensors have the same deployment parameters and have the same sensing coverage (R_c) and the same communication range (R_s). The former helps in the detection of the designed event. The latter is introduced for the connectivity of nodes with their neighborhoods. Thus, depending on the communication range the neighborhood of a node is defined. We say that the probability of communication between two nodes varies inversely with the distance between them.

We evaluate the proposed approach via simulations (NS2). Our topology is a square with total area *A* = 20 x 20, A_i is the area of the Voronoi polygon for a sensor for a full sensing coverage and a full connectivity defined as [13]:

$$A_s(Cov, Con) = min(2R_s^2, R_c^2) \quad (2)$$

Twenty seconds is considered as the simulation time. It should be mentioned that we assume that all links are bi-directional during simulation. The sensing range is set to 2 while the communication range is set to 4 and the maximum number of neighbors (N_e) for each deployed sensor is set to 4. Thus, each sensor must have a number of neighbors less than N_e (fixed to 4 in our case).

We assume that the algorithm stops when the coverage *C* is greater than 1, we define *C* as the ratio of the union of all areas covered by each node and the area of the entire Region Of Interest (ROI) [9]. The coverage *C* helps to ensure that the region is entirely covered by the sensors when deploying them. In this work, we try to take into account the overlapping zone between nodes when calculating the area covered by the sensors. Each node is characterized by a covered area which is defined as the circular area within its sensing radius R_s.

$$C = \frac{\cup_{i=1..N} A_i}{A} \quad (3)$$

Where A_i is the area covered by the *i*th node,

N is the total number of nodes,

A stands for the area of ROI.

Our approach is used in order to determine a better methodology for sensor placement problem in a WSN. As for results, the number of sensors in the region of interest varies from 51 to 57 providing full sensing coverage and full network connectivity. We can observe that our approach outperforms Max_Avg_Cov [14] [15] and MIN-MISS [15] [16]. It is important to point out that in this work, we do not take into account obstacles in the ROI.

V. Conclusion

In this paper, the sensor placement problem is studied. We have formulated a new strategy for the initial sensor placement problem. We start with a random distribution of nodes with constraints of coverage and connectivity. As part of our future work, we would extend this result to a case when there are obstacles in the ROI. Thus, we assume that the above results are practical and can be used in actual sensor network design.

Acknowledgment

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

References

[1] F. Lin and P.-L. Chiu, "A near-optimal sensor placement algorithm to achieve complete coverage-discrimination in sensor networks," Communications Letters, IEEE 9.1, vol. 9, 2005, pp. 43–45.

[2] L. T.-M. E. C. Ruben Leal, Jose Aguilar and A. Rios, "An Approach for Diagnosability Analysis and Sensor Placement for Continuous Processes Based on Evolutionary Algorithms and Analytical Redundancy," Applied Mathematical Sciences, vol. 9, no 43, 2015, pp. 2125–2146.

[3] M. Z. A. Bhuiyan, G. Wang, J. Cao, and J. Wu, "Sensor placement with multiple objectives for structural health monitoring," ACM Transactions on Sensor Networks (TOSN), vol. 10, no 4, 2014, pp. 68–113.

[4] K. A. Izzet F. Senturk and F. Senel, "An effective and scalable connectivity restoration heuristic for mobile sensor/actor networks," Global Communications Conference (GLOBECOM), 2012 IEEE, 2012, pp. 518–523.

[5] F. Guerriero, A. Violi, E. Natalizio, V. Loscri, and C. Costanzo, "Modelling and solving optimal placement problems in wireless sensor networks," Applied Mathematical Modelling, vol. 35, no 1, 2011, pp. 230–241.

[6] E. Flushing and G. Di Caro, "Exploiting synergies between exact and heuristic methods in optimization: an application to the relay placement problem in wireless sensor networks," Bio-Inspired Models of Network, Information, and Computing Systems, vol. 134, 2014, pp. 250–265.

[7] J. O'Rourke, Ed., Art Gallery Theorems and Algorithms. Oxford University Press, 1987.

[8] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," Ad Hoc Networks, vol. 6, no 4, 2008, pp. 621–655.

[9] N. Heo and P. K. Varshney, "An intelligent deployment and clustering algorithm for a distributed mobile sensor network," Systems, Man and Cybernetics, 2003. IEEE International Conference on, vol. 5, 2003, pp. 4576–4581.

[10] P. Cheng, C.-N. Chuah, and X. Liu, "Energy-aware node placement in wireless sensor networks," Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, vol. 5, 2004, pp. 3210–3214.

[11] S. S. Dhillon, K. Chakrabarty, and S. Iyengar, "Sensor placement for grid coverage under imprecise detections," Information Fusion, 2002. Proceedings of the Fifth International Conference, vol. 2, 2002, pp. 1581–1587.

[12] S. Adlakha and M. Srivastava, "Critical density thresholds for coverage in wireless sensor networks," Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, vol. 3, 2003, pp. 1615–1620.

[13] Y. Wang, Y. Zhang, J. Liu, and R. Bhandari, "Coverage, Connectivity, and Deployment in Wireless Sensor Networks," Recent Development in Wireless Sensor and Ad-hoc Networks, 2015, pp. 25–44.

[14] S. S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," vol. 3. IEEE, 2003, pp. 1609–1614.

[15] S. M. Reda, M. Abdelhamid, O. Latifa, and A. Amar, "Efficient uncertainty-aware deployment algorithms for wireless sensor networks," in Wireless Communications and Networking Conference (WCNC), 2012 IEEE. IEEE, 2012, pp. 2163–2167.

[16] Y. Zou and K. Chakrabarty, "Uncertainty-aware sensor deployment algorithms for surveillance applications," in Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE, vol. 5. IEEE, 2003, pp. 2972–2976.

 ISBN: 978-1-61208-438-1

Dynamic Symbolic Execution using Eclipse CDT

Andreas Ibing

Chair for IT Security
TU München
Boltzmannstrasse 3, 85748 Garching, Germany
Email: andreas.ibing@tum.de

***Abstract*—Finding software bugs before deployment is essential to achieve software safety and security. The achievable code coverage and input coverage with manual test suite development at reasonable cost is limited. Therefore, complementary automated methods for bug detection are of interest. This paper describes automated context-sensitive detection of software bugs with dynamic symbolic execution. The program under test is executed in a debugger, and program execution is automatically driven into all program paths that are satisfiable with any program input. Program input and dependent data are treated as symbolic variables. Dynamic analysis and consistent partial static analysis are combined to automatically detect both input-dependent and input-independent bugs. The implementation is a plug-in extension of the Eclipse C/C++ development tools. It uses Eclipse's code analysis framework, its debugger services framework and a Satisfiability Modulo Theories automated theorem prover. The resulting dynamic symbolic execution engine allows for consistent partially concrete program execution. Compared to static symbolic execution, it transfers as much work as possible to concrete execution in a debugger, without losing bug detection accuracy. The engine is evaluated in terms of bug detection accuracy and runtime on buffer overflow test cases from the Juliet test suite for program analyzers.**

Keywords–Testing; Program analysis; Symbolic execution.

I. INTRODUCTION

Software bugs in general are context-sensitive, so that a context-sensitive algorithm is needed for accurate detection. Symbolic execution [1] is an approach to automated context-sensitive program analysis. In can be applied as static analysis, in the sense of symbolic interpretation. Program input is treated as symbolic variables, and operations on variables yield logic equations. The satisfiability of program paths and bug condition satisfiability are decided with an automated theorem prover (constraint solver). Symbolic execution in principle is applicable to all levels of software, i.e., models, source code, intermediate code and binaries.

As code execution is in general faster than interpretation, symbolic execution has also been applied as dynamic analysis [2]. When certain software parts can not practically be treated by static symbolic execution, this offers a way for dynamic analysis by concretizing symbolic variables as approximation, without introducing false positive bug detections (although this leads to false negative detections) [2]. The software parts which are treated symbolically (static) and which concretely (dynamic) can be made selectable for a tool user in the sense of selective symbolic execution [3]. A more detailed overview of available symbolic execution tools and applications is available in [4][5].

Symbolic execution relies on an automated theorem prover as logic backend. The current state in automated theorem proving are Satisfiability Modulo Theories (SMT) solver [6]. An example state of the art solver is described in [7]. A standard interface to SMT solvers has been defined with the SMTlib [8].

The different types of softwate bugs are classified in the common weakness enumeration (CWE) [9]. Examples are stack based buffer overflows with number CWE-121 and heap based buffer overflows as CWE-122.

For the evaluation of automated software analyzers, test suites have been developed. The evaluation criteria are the number of false positive and false negative bug detections and the needed run time. Currently, the most comprehensive test suite for C/C++ is the Juliet suite [10]. It systematically tests the correct detection of different common weakness types (as baseline bugs) in combination with different data and control flow variants which cover the available programming language grammar constructs. The suite contains both 'good' (without bug) and 'bad' (including a bug) functions in order to measure false positive and false negative detections. The maximum bug context depth of a flow variant are five functions in five different source files.

This paper describes a dynamic symbolic execution approach, that combines static and dynamic checks in order to detect both input-dependent and input-independent bugs. It builds upon an existing purely static symbolic execution engine described in [11]. The work at hand differs in that most of the work is transferred to a debugger, which additionally allows for consistent partially concrete program execution. The debugger is automatically driven into all executable program paths, and bugs are detected both during concrete execution and during symbolic interpretation.

The remainder of this paper is organized as follows: Section III gives an overview of the algorithm which is used to traverse the program execution tree. Section IV describes the implementation which extends the Eclipse C/C++ development tools (CDT). In Section V the achieved bug detection accuracy and run times are evaluated with buffer overflow tests from the Juliet suite. Section II gives an overview of related work, and Section VI discusses the obtained results.

II. RELATED WORK

There is a large body of work on symbolic execution available which spans over 30 years [12]. Dynamic symbolic execution is described in [2][13][14][15]. To reduce complexity and increase analysis speed, as many variables as possible

 ISBN: 978-1-61208-438-1

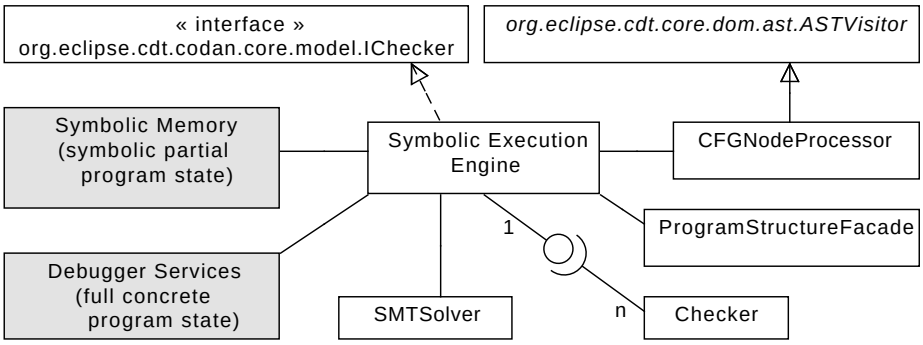


Figure 1. Architecture overview.

are regarded as concrete values. Only variables which depend on program input are modelled as symbolic. The analysis runs dynamically as long as all parameters are concrete, and equation systems for the solver are smaller. In [13], single-stepping is used together with a check whether a symbolic variable is contained in the respective statement. Selective symbolic execution is presented in [3]. It allows to choose which parts of a system are executed symbolically. It uses the qemu virtual machine monitor with an LLVM backend and runs the engine from [15] on it.

The presented approach differs in several aspects. One point is the tight IDE integration which might enable synergies with other Eclipse tools. Static and dynamic checks are combined in order to also detect bugs which are not input dependent. Breakpoints are set adaptively to interprete as few code lines as possible without degrading bug detection accuracy, and complex dependencies between symbolic variables are tracked.

III. Algorithm

The algorithm is basically depth-first search. It is used to traverse the tree of satisfiable paths through the program under test, which is commonly called the program execution tree.

Execution of a program path changes between concrete execution in the debugger and symbolic interpretation. Debugger breakpoints are used to switch from concrete execution to symbolic interpretation. The debugger contains a full conrete program state. The interpreter contains the partial variable set, which needs to be symbolic, i.e., the values are logic formulas. The full concrete program state and the partial symbolic state are consistent, i.e., the concrete state satisfies the symbolic state constraints.

C programs interact with their environment through functions from the C standard library (libc). The symbolic execution engine can trace input and can determine a program path by forcing corresponding input.

Certain library functions are defined a-priori to have symbolic return variables. Correspondingly, initial breakpoints are inserted at call locations to the specified functions. The program argument vector is also treated as symbolic, i.e., breakpoints are set at locations where it is accessed. Breakpoints are inserted only in the source files of interest.

For the first execution path, the engine traces program input. In case of blocking functions, a direct return with a valid error return value is forced. The argument vector is initially set to be empty.

Concrete variables may become symbolic, i.e., when they are assigned a formula. Then corresponding breakpoints at access to the new symbolic variable are set. Symbolic variables may become concrete, i.e., when they are assigned a concrete value. Then, the corresponding breakpoints are removed.

Bug detection in concretely executed program parts is performed with run-time checks, i.e., dynamic analysis. Bug detection in interpreted parts, i.e., input dependent, is performed using satisfiability queries to the solver.

After reaching the program end, program input is automatically generated for the next path to explore. The solver and its model generation functionality are used to generate concrete input values, which are forced in the next program run. The input determines that the next program run will take a different branch, according to depth-first traversal of the execution tree.

IV. Implementation as Eclipse CDT plug-in

A. Architecture Overview

An overview of the architecture is given in Figure 1 as class diagram. The symbolic execution engine performs tree-based interpretation [16] for program locations which use symbolic variables. The engine can be started from the CDT GUI through an extension point provided by the code analysis framework. The syntax files of interest are parsed into abstract syntax trees (AST) with CDT's C/C++ parser. Translation into logic equations is performed according to the visitor pattern [17] using CDT's `ASTVisitor` class. The interpreter has a partial symbolic memory store which contains the symbolic variables (global memory and function space stack). For the rest, CDT's debugger services framework is used. A full concrete program state is available in the debugger. For the detection of input dependent bugs, the engine provides a checker interface. Through this interface, checker classes can register for triggers and query context information, which is necessary for the corresponding solver satisfiability checks.

B. Short review of CDT's code analysis framework

The code analysis framework (Codan [18]) is a part of CDT. It provides GUI integration for checker configuration

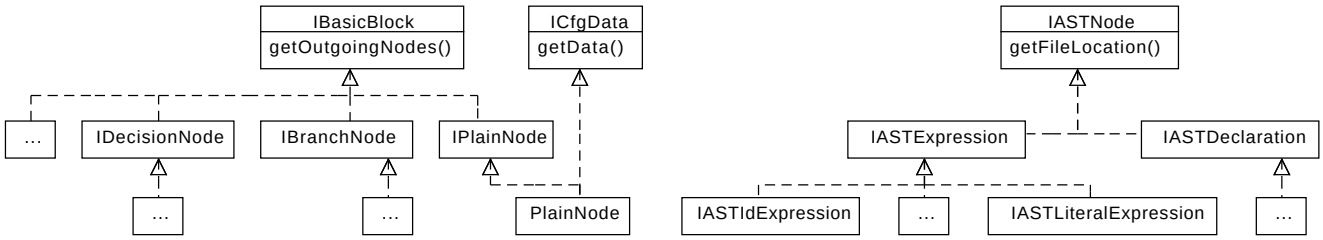


Figure 2. Important data structures provided by CDT and its code analysis framework for CFG (left) and AST (right).

and result presentation using Eclipse's marker framework. It further provides a control flow graph (CFG) builder. Figure 2 illustrates important data structures for AST and CFG. There are different CFG node types for plain nodes, decision nodes, branch nodes, jump nodes etc. A CFG node (depending on the type) typically includes a reference to the corresponding AST subtree. Static program analysis normally evaluates paths through CFGs [11]. Tree-based interpretation means that for a CFG node the referenced AST subtree is interpreted. An AST node provides a reference to the corresponding source location.

C. Short review of CDT's debugger services framework

Debuggers typically feature a machine interface (MI) to ease the development of graphical debugger frontends. CDT includes a debugger services framework (DSF) [19], which is an abstraction layer over debuggers' machine interfaces. DSF provides a set of asynchronous services. The main service interfaces are illustrated in Figure 3. They are used to control dynamic execution with the debugger (`IMIRunControl`) and to insert breakpoints (`IBreakpoints`). The current program location and variables can be queried. This comprises local (`IStack`) and global variables (`IExpressions`).

D. Partial symbolic interpretation

The debugger stops at breakpoints or when it receives a signal. The symbolic execution engine then switches to symbolic interpretation and tries to resolve the respective CFG node and AST subtree. The source location (file and line number) can be obtained from CDT (CDT's `CSourceLocator`). In order to enable CFG node resolution, a location map for the source files of interest is pre-computed before analysis start. The resolved CFG node is then followed to its AST subtree, which is symbolically interpreted. Needed concrete values are queried from the debugger The translation into logic equations uses the SMTlib sublogic of arrays, uninterpreted functions and bit-vectors (AUFBV).

E. Input-dependent branches

The debugger only breaks at a decision when the decision contains a symbolic variable (input-dependent branch). Possible branch targets (CFG branch nodes and their children) are obtained as children of the corresponding decision node. The debugger is commanded to step, and the taken branch is identified through the newly resolved CFG node. The branch constraint is formulated as symbolic formula. Branch constraints need to be remembered to enable input generation for the next execution path. If there is already a breakpoint set for the source location after stepping, then this location is also symbolically interpreted. Otherwise the debugger is commanded to resume execution.

F. Implementation of read/write watchpoints

Concrete execution must be breaked at read and write accesses to symbolic variables. This means conceptually that a very large number of read/write watchpoints is needed. Software watchpoints would severely slow down debugger execution and in general are only available as write watchpoints, not read watchpoints [20]. Hardware watchpoints can also not be used, since standard processors only support a handful of them. The implementation therefore uses normal software line breakpoints and determines the relevant locations using the available source code. To this end, a map of language bindings is pre-computed before analysis. The map contains AST names with references to the corresponding source file locations. When a variable becomes symbolic, the corresponding breakpoints are inserted (through DSF's `IBreakpoints` interface, Figure 3).

For accurate bug detection it is additionally necessary to trace pointer targets. Pointer assignments must be traced because there may be pointers to a target when the target becomes symbolic. The initial breakpoints therefore also include breaks on all pointer assignments (a pointer target map is then updated accordingly). This requirement is illustrated in the experiments section with Figure 5.

G. Controlling program input

In order to trace and force program input, the debugger is set to break at calls to functions from the standard library. Because the `libc` contains several functions for which the debugger cannot step into or break inside (e.g., functions that directly access the virtual dynamically shared object), these functions are wrapped. For non-blocking functions on the first program path, the `step into` and `finish` debugger commands are used to trace the function's return value. For blocking functions or on later program paths, the program input is set to the pre-determined value (or solver-generated value respectively) with `step into` and `return` debugger commands.

H. Bug detection

1) During concrete execution: Instrumentation and dynamic analysis are used to detect bugs during concrete execution. This paper uses the example of buffer overflow detection. To dynamically detect buffer overflows, the available address sanitizer from [21] is used. In [21], it is reported that the instrumentation slows down execution by about 73% and

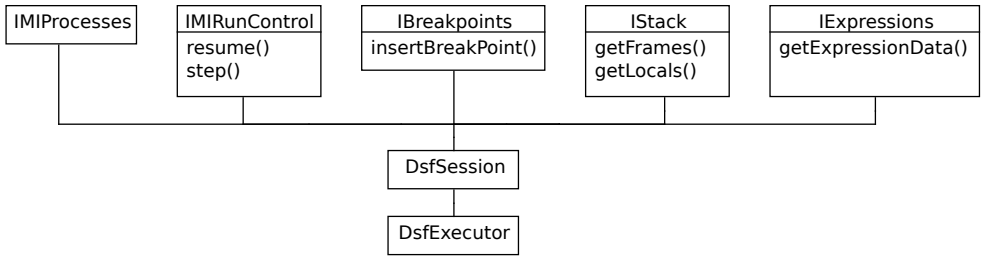


Figure 3. Main interfaces of CDT's debugger service framework.

```
1  void CWE121_memcpy_01_bad() {
2    charvoid cv_struct;
3    cv_struct.y = (void *)SRC_STR;
4    // FLAW: overwrite the pointer y
5    memcpy(cv_struct.x, SRC_STR,
6      sizeof(cv_struct));
7    cv_struct.x[(sizeof(cv_struct.x)/
8      sizeof(char))-1]='\0';
9    printLine((char *)cv_struct.x);
10   printLine((char *)cv_struct.y);
11 }
```

Figure 4. Buffer overflow detection during concrete execution. Example from [22]

increases memory usage about 3.4 times. The source code under test is compiled and statically linked with the address sanitizer library. A breakpoint is set on the address sanitizer error report function. In case the debugger breaks at this location, the bug is localized by following the call stack back into the source files of interest.

2) During symbolic interpretation: Input dependent bugs are detected with solver queries during symbolic interpretation. For buffer overflows, the bounds checker is triggered during interpretation of array subscript expressions and pointer dereferences, when the index expression or pointer offset are symbolic. The checker then queries the solver whether index/offset can be smaller than zero or lerger than the buffersize.

I. Input generation

The parts of an execution path that are symbolically interpreted can be denoted as symbolic execution path. To generate input for the next execution path, the symbolic execution path is backtracked to the last decision node. For any unvisited child branch nodes, satisfiability of the backtracked path constraint together with the respective branch constraint is checked using the solver. If the constraints are satisfiable, corresponding input values are generated using the solver's model generation functionality (`get-model` command). If the constraints are not satisfiable, first the unvisited branch siblings are tested, then the symbolic execution path is further backtracked. Traversal of the symbolic execution tree (and therefore also the execution tree) is complete when further backtracking is not possible.

```
void CWE121_fgets_32_bad() {
  int data = -1;
  int *data_ptr1 = &data;
  int *data_ptr2 = &data;
  { int data = *data_ptr1;
    char input_buf[CHAR_ARRAY_SIZE] = "";
    if (fgets(input_buf, CHAR_ARRAY_SIZE,
      stdin) != NULL)
    {  data = atoi(input_buf); }
    else
    { printLine("fgets() failed."); }
    *data_ptr1 = data;
  }
  { int data = *data_ptr2;
    int buffer[10] = { 0 };
    if (data >= 0) {
      // FLAW: possible buffer overflow:
      buffer[data] = 1;
      for(int i = 0; i < 10; i++)
      { printIntLine(buffer[i]); }
    }
    else
      { printLine("ERROR: out of bounds");}
  }
}
```

Figure 5. Buffer overflow detection during symbolic interpretation. Example from [22]

V. Experiments

A. Test cases and test set-up

The used test set consists of 58 small buffer overflow test programs from the Juliet suite [22]. The test programs are analysed with the Eclipse plug-in, using Eclipse version 4.5 (CDT 8.8.0) on a i7-4650U CPU, on 64-bit Linux kernel 3.16.0 with GNU debugger gdb version 7.7.1 [20]. The test set contains stack based buffer overflows with `memcpy()` (19 test programs) and with `fgets()` (39 test programs). The test programs cover all 39 Juliet flow variants for C. The results are illustrated in Figure 6. The figure contains run-times for correct detection only (no false positives or false negatives). Flow variants in [22] are not numbered consecutively in order to allow for later insertions.

The test set contains bugs which are detected during concrete execution (`memcpy()`) and with input-depending branching (e.g., flow variant 12). It also contains input-depending bugs (with `fgets()`), which are detected during symbolic interpretation using the solver (bug condition satisfiability check).

 ISBN: 978-1-61208-438-1

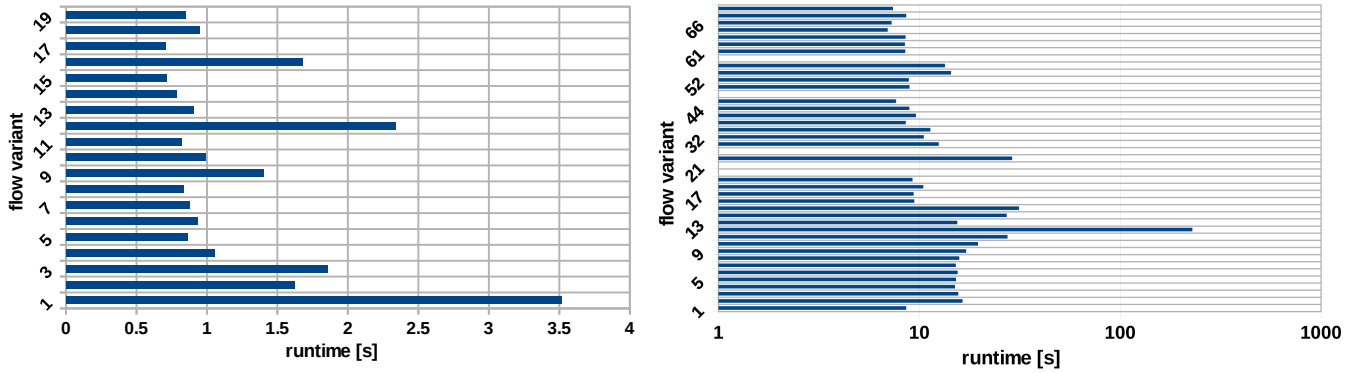


Figure 6. Analysis run-times for correct bug detection with dynamic symbolic execution.

B. Detection accuracy

The functionality is illustrated with two source code listings from [22]. The listings have been slightly modified to be shorter.

Figure 4 shows an example of bug detection during concrete execution in the debugger. The 'bad' function contains a baseline bug (simplest flow variant) with `memcpy` in line 6. The bug is that the size of the complete struct is used where only the size of a contained array is meant. The debugger breaks on the address sanitizer's error reporting function and the bug is correctly localized.

Figure 5 illustrates the need for pointer tracing with flow variant 32 ('data flow using two pointers to the same value within the same function' [22]). The 'bad' function contains three variables `data` (declared in lines 2, 5 and 14). An initial breakpoint is set on the `fgets` function call in line 7. The second `data` variable becomes symbolic in line 9 due to the `atoi` library call, so that a breakpoint is set in line 12 (read access to this `data`). With an assignment through pointer dereference in line 12 the first `data` variable (from line 2) becomes symbolic. This would have been missed without tracing the pointer targets (here the pointer assignment in line 3). In line 12, also `data_ptr2` becomes symbolic, because it points to the now symbolic first `data`. Therefore also `data_ptr2` is watched, i.e., a breakpoint is set on line 14. In line 14, the third `data` variable becomes symbolic and is watched, so that the debugger breaks on lines 16 (where a constraint is collected) and 18. In line 18, solver bounds checks are trigged for the array subscript expression, and the buffer overflow is detected because the solver decides that the index expression might be larger than the buffer size.

Bug detection during concrete execution depends on the available instrumentation and run-time checks. The address sanitizer used as example first misses a buffer overflow with flow variant 9 ('control flow depending on global variables'), but then detects it through reception of a segmentation fault signal from the operating system.

The bugs are correctly detected for all flow variants apart from variant 21 ('control flow controlled by value of a static global variable'). For this variant, the CFG builder misclassifies a branch node as dead node. This leads to missing program paths in the analysis and consequently to a false negative detection.

C. Speed

Figure 6 shows analysis runtimes for the buffer overflow test cases with `memcpy` on the left, and with `fgets` on the right. The vertical axis shows the numbering of data and control flow variants from Juliet [22]. The horizontal axis shows the measured runtime (wall-clock time) in seconds. The tool needs about 1-2s for each of the `memcpy` test cases, and about 20s for each of the `fgets` test cases. An exception is the `fgets` test with flow variant 12. It contains quite a few concatenated decisions for which both branches are satisfiable. This leads to exponential path explosion. In addition, the debugger execution is restarted many times from the program start. This means that overlapping start paths are re-executed redundantly.

VI. Conclusion and Future Work

This paper presents an Eclipse CDT plug-in for automated bug detection with dynamic symbolic execution. Software bugs are detected with combined static and dynamic checks. As much work as possible is transferred to a debugger, whose execution is driven into all executable program paths. The presented approach is applicable with native and with cross compilation. It can be applied, e.g., with the qemu virtual machine monitor which contains a gdb server for the guest virtual machine. The current implementation suffers from the path explosion problem, i.e., the number of satisfiable paths in general grows exponentially with program length. Ongoing work therefore aims to improve the scaling behaviour by implementing ways to detect and prune program paths, on which the detection of new bugs is not possible.

Acknowledgement

This work was funded by the German Ministry for Education and Research (BMBF) under grant 01IS13020.

References

[1] J. King, "Symbolic execution and program testing," Communications of the ACM, vol. 19, no. 7, 1976, pp. 385–394.

[2] P. Godefroid, N. Klarlund, and K. Sen, "DART: Directed automated random testing," in Conference on Programming Language Design and Implementation, 2005, pp. 213–223.

[3] V. Chipounov, V. Kuznetsov, and G. Candea, "S2E: A platform for in-vivo multi-path analysis of software systems," in Int. Conf. Architectural Support for Programming Languages and Operating Systems, 2011, pp. 265–278.

 ISBN: 978-1-61208-438-1

[4] C. Cadar, K. Sen, P. Godefroid, N. Tillmann, S. Khurshid, W. Visser, and C. Pasareanu, “Symbolic execution for software testing in practice – preliminary assessment,” in Int. Conf. Software Eng., 2011, pp. 1066–1071.

[5] C. Pasareanu and W. Visser, “A survey of new trends in symbolic execution for software testing and analysis,” Int. J. Software Tools Technology Transfer, vol. 11, 2009, pp. 339–353.

[6] L. deMoura and N. Bjorner, “Satisfiability modulo theories: Introduction and applications,” Communications of the ACM, vol. 54, no. 9, 2011, pp. 69–77.

[7] ——, “Z3: An efficient SMT solver,” in Tools and Algorithms for the Construction and Analysis of Systems (TACAS), 2008, pp. 337–340.

[8] C. Barrett, A. Stump, and C. Tinelli, “The SMT-LIB standard version 2.0,” in Int. Workshop Satisfiability Modulo Theories, 2010.

[9] R. Martin, S. Barnum, and S. Christey, “Being explicit about security weaknesses,” CrossTalk The Journal of Defense Software Engineering, vol. 20, 3 2007, pp. 4–8.

[10] T. Boland and P. Black, “Juliet 1.1 C/C++ and Java test suite,” IEEE Computer, vol. 45, no. 10, 2012, pp. 88–90.

[11] A. Ibing, “Symbolic execution based automated static bug detection for Eclipse CDT,” Int. J. Advances in Security, vol. 1&2, 2015, pp. 48–59.

[12] C. Cadar and K. Sen, “Symbolic execution for software testing: Three decades later,” Communications of the ACM, vol. 56, no. 2, 2013, pp. 82–90.

[13] C. Cadar, V. Ganesh, P. Pawlowski, D. Dill, and D. Engler, “EXE: Automatically generating inputs of death,” in 13th ACM Conference on Computer and Communications Security (CCS), 2006, pp. 322–335.

[14] P. Godefroid, M. Levin, and D. Molnar, “Automated whitebox fuzz testing,” in Network and Distributed System Security Symp. (NDSS), 2008, pp. 151–166.

[15] C. Cadar, D. Dunbar, and D. Engler, “KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs,” in USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2008, pp. 209–224.

[16] T. Parr, Language Implementation Patterns. Pragmatic Bookshelf, 2010.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[18] E. Laskavaia, “Codan- a C/C++ code analysis framework for CDT,” in EclipseCon, 2015.

[19] P. Piech, T. Williams, F. Chouinard, and R. Rohrbach, “Implementing a debugger using the DSF framework,” in EclipseCon, 2008.

[20] R. Stallman, R. Pesch, and S. Shebs, “Debugging with gdb,” 2011, [retrieved: Sept., 2015]. [Online]. Available: http://sourceware.org/gdb/current/onlinedocs/gdb.html

[21] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, “Address-Sanitizer: A fast address sanity checker,” in USENIX Annual Technical Conference, 2012, pp. 28–28.

[22] Juliet Test Suite v1.2 for C/C++, United States National Security Agency, Center for Assured Software, May 2013, [retrieved: Sept., 2015]. [Online]. Available: http://samate.nist.gov/SARD/testsuites/juliet/Juliet_Test_Suite_v1.2_for_C_Cpp.zip

 ISBN: 978-1-61208-438-1

Evaluating the Usability of Mobile Instant Messaging Apps on iOS Devices

Sergio Caro-Alvaro, Antonio Garcia-Cabot, Eva Garcia-Lopez, Luis de-Marcos, Jose-Javier Martinez-Herráiz
Computer Science Department
University of Alcala
Alcala de Henares, Spain
sergio.caro@edu.uah.es; a.garciac@uah.es; eva.garcial@uah.es; luis.demarcos@uah.es; josej.martinez@uah.es

***Abstract*— Instant messaging apps are experiencing a significant upturn in recent years in mobile devices. This paper shows the results of applying a systematic evaluation of these applications on iOS platform that was performed to identify their main usability issues. As a result of this evaluation some guidelines for improving the usability of these applications are proposed, such as a carefully designing the interface or not exceeding more than eight (and preferably not more than six) interactions to perform the main tasks. The results and the guidelines proposed will help in the future to create more effective mobile applications for instant messaging.**

Keywords- Instant messaging; mobile usability; keystroke level modeling; mobile heuristic evaluation

I. INTRODUCTION

The increased use of mobile devices [1] has led to the number of applications (apps) available in mobile markets has also increased significantly in recent years, such as instant messaging (IM) apps, which have become ubiquitous in contemporary society. To increase the chances of an app to be chosen by users among many others it is essential that it has a good usability. It is important to study usability in desktop applications, but it is even more important to study it in mobile apps because mobile devices have some limitations when compared to personal computers (PC) [2] [3], such as small-sized screens, limited input mechanisms, battery life, etc. These characteristics make necessary studying usability for mobile devices separately from usability for PCs.

A mechanism for systematic evaluation (also called protocol) was created by Martin et al. [4] for studying usability in existing mobile apps, and consists of five steps: (1) Identify all potentially relevant applications, (2) remove light or old versions of each application, (3) identify the primary operating functions and exclude all applications that do not offer this functionality, (4) identify all secondary functionality and (5) test the main functionalities using the following methods: Keystroke-Level Modeling (KLM) for estimating the time taken to complete each task to provide a measure of efficiency of the applications [5] [6] and Mobile Usability Heuristics (MUH) for identifying more usability problems using a usability heuristic evaluation.

Since mobile IM apps are becoming widely used in recent years, it is especially important to study the most common usability problems in this kind of apps, in order to get some guidelines or good practices for developers of mobile IM apps. This paper shows a systematic evaluation of instant messaging apps and the results obtained during this evaluation. Finally, some recommendations are proposed from a viewpoint of mobile usability.

The paper is organized as follows: Section 2 shows the evaluation carried out and the results obtained in the systematic evaluation. Section 3 presents a discussion of the results with previous work and, finally, Section 4 explains the recommendations and conclusions obtained from the results.

II. EVALUATION AND RESULTS

This section shows the systematic evaluation carried out and the different results obtained in the steps. iOS platform (an iPhone 4) was used along the evaluation steps.

A. *Steps 1 to 4*

In the first step, potential and relevant applications available in the iOS app store were identified. The "instant messaging" term was used to search in the app market. As a result, 243 applications were classified as potential applications.

In the second step, the applications that were not fully functional (i.e., demos, lite or trials) were removed from the list of potential applications. In all, 20 applications (8%) were removed from the initial list.

An application can be considered as instant messaging when it meets all the main functionalities, which were defined in Step 3 as follows:

- Task 1 (T1). Send an instant message to a specific contact ([7]).
- Task 2 (T2). Read and reply an incoming message ([7]).
- Task 3 (T3). Add a contact ([7]).
- Task 4 (T4). Delete/Block a contact (derived from [8] and [9]).
- Task 5 (T5). Delete chats ([10]).

Once the main functionalities were detected and defined, the applications that did not meet all these requirements were discarded. As a result, only 39 (18%) applications met the main functionalities to be considered as IM applications.

In the fourth step, it was necessary to discover the secondary functionalities on the applications selected in the previous step (11 apps were discarded because they ran

 ISBN: 978-1-61208-438-1

anomaly with unrecoverable errors). The most common secondary functionalities detected were including a user profile avatar (74.36%), sending pictures (66.67%) and sending videos (64.10%), among others.

B. Step 5-A: Keystroke-Level Modeling

In this step, the remaining applications (28 apps in total) were reviewed in order to count the number of interactions (KLM) required to perform each of the main functionalities established in Step 3.

TABLE I. TOP10 KLM RESULTS: NUMBER OF INTERACTIONS REQUIRED FOR COMPLETING THE TASKS

App	v.	T1	T2	T3	T4	T5	Total
Surespot encrypted messenger	6.00	5	5	3	4	4	21
Hike messenger	2.5.1	5	5	5	5	4	24
HushHush App	1.0.3	6	6	4	4	4	24
Hiapp Messenger	1.0.6	6	6	5	5	4	26
Kik Messenger	7.2.1	5	5	5	6	5	26
Touch	3.4.4	7	5	5	5	4	26
WhatsApp Messenger	2.11.8	5	6	5	5	6	27
BBM	2.1.1.64	6	5	7	6	4	28
iTorChat	1.0	7	6	5	5	5	28
XMS	2.31	6	6	6	6	4	28
Mean		**6.54**	**5.75**	**6.25**	**5.96**	**5.36**	**29.85**

Table 1 only shows the top 10 with the fewest interactions; the minimum number of interactions for completing all tasks was 21 (Surespot encrypted messenger) and the app with maximum (39 interactions) was Spotbros app, but obviously it is not shown in Table 1. An average of 6 interactions for each task can also be observed.

In order to send a new message (task 1), the apps with fewer interactions (5 interactions) obtained these results by showing the keyboard automatically, although only 6 apps have this feature (WhatsApp, Tuenti, IM+ Pro7, Hike, Surespot and Kik).

All analyzed apps required between 4 and 6 interactions to reply to a given message (task 2), except Spotbros, which required 8 interactions because chats were shown only when a button was pressed. The similarity in the number of interactions is because almost all applications had a section that contained the active chats grouped.

The most variations were observed in task 3 (adding a contact): from 3 interactions (Surespot encrypted messenger) to 10 interactions (Tuenti and Spotbros). This is because some applications (11 of all apps analyzed) used the agenda of the mobile device and others used their own contact list, causing alternative implementations of this process, thus requiring extra data in some cases.

Finally, for task 5 (deleting a chat) most of the examined apps required between 4 and 6 keystrokes, due to the similar implementation of the process.

For the next step (5.B. heuristic evaluation), not all apps were selected to continue in the process. As in previous studies [11]-[13], the four applications with fewer interactions were selected for the next step. In this case, applications with the same number of interactions were considered as one. Therefore, 7 applications in total were selected: Surespot (21 interactions), Hike Messenger and HushHushApp (both 24 interactions), Kik Messenger, Hiapp Messenger and Touch (26) and WhatsApp Messenger (27).

C. Step 5-B: Mobile Usability Heuristics

In this step, the mobile usability evaluation using heuristics was performed. Six (6) experts carried out the evaluation with the 7 applications selected in the previous step. As Bastien [14] and Hwang and Salvendy [15] indicated, from 5 to 10 users participating in the evaluation is enough to detect at least 80% of the usability issues in software. The Mobile Heuristic Evaluation (MHE) method [2] is based on a study in which each expert checks whether each application meets or not a set of directives for usability, which includes directives about usability features of the application that the expert has to answer, expressing their opinion with a numeric value from 0 to 4 (where 0 indicates that there is no problem, and 4 indicates a catastrophic problem), which is known as Nielsen's five-point Severity Ranking Scale [16], and they also had to justify their scores. The eight heuristics used were [2]: A (visibility of system status and losability / findability of the mobile device), B (match between system and the real world), C (consistency and mapping), D (good ergonomics and minimalist design), E (ease of input, screen readability and glancability), F (flexibility, efficiency of use and personalization), G (aesthetic, privacy and social conventions) and H (realistic error management).

The results of the evaluation are shown in Table 2. Applications with lower values (i.e., more usable applications) had mostly cosmetic problems (small obstacles) or no problems. On the other hand, applications with higher values had mainly minor and major problems, obstacles that affect the functionality of the application in a regular use.

TABLE II. RESULTS OF HEURISTIC EVALUATION ON APPLICATIONS

Mobile usability heuristics results									
App	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**Total**
#1[a]	0.08	0.28	0.92	0.67	0.13	0.42	1.58	0.00	**4.08**
#2[b]	0.92	0.61	0.67	1.00	0.23	1.17	0.50	0.39	**5.48**
#3[c]	0.00	1.17	1.67	1.42	0.70	0.50	1.25	0.89	**7.59**
#4[d]	2.54	2.17	1.75	0.58	1.03	1.50	0.50	0.61	**10.69**
#5[e]	2.63	1.61	2.08	1.17	1.33	1.25	1.67	1.00	**12.74**
#6[f]	0.00	2.00	1.67	1.92	1.07	2.08	3.17	0.89	**12.79**
#7[g]	1.00	2.39	1.75	2.08	1.57	0.67	2.33	1.44	**13.23**
Mean	**1.02**	**1.46**	**1.50**	**1.26**	**0.87**	**1.08**	**1.57**	**0.75**	**9.51**

a. WhatsApp Messenger
b. HushHushApp
c. Hiapp Messenger

 ISBN: 978-1-61208-438-1

d. Surespot encrypted messenger
e. Kik Messenger
f. Touch
g. Hike Messenger

The mean of heuristics show that G heuristic is the one that got the worst score (Table 2), due to a (generally) bad interface design and the lack of privacy and security information. The second worst heuristic was C heuristic, mainly because in some apps some objects were not expected on the interface. On the other hand, H heuristic got the best results, thanks to the ease on editing incorrect inputs and also the ease on recovering from errors. The E heuristic was the second best rated because of the ease on entering numbers, as well as it shows a back button on the screens and (mainly) the ease on navigation through the screens.

Finally, it is worth mentioning that a low number of interactions does not necessarily imply that the application has not usability problems. For instance, Hike messenger had 24 interactions and was the second best app on KLM results, whereas it was the worst app according to the MHE results. This implies that both techniques should be applied in order to evaluate the usability of an application.

III. DISCUSSION

In this section, the results and analysis carried out will be discussed. Firstly, we could compare the results obtained with those from two similar studies performed using a similar method: one for spreadsheet apps [13] and another one for diabetes management apps [12].

Step 1 (potentially relevant applications) produced 23 spreadsheet apps, 231 diabetes apps and we found 243 IM apps. The low number of apps in the spreadsheet study may be due to a more concrete term. Step 2 (delete light or old versions) discarded 9 (4.05%) diabetes apps and we discarded 20 (8.97%) IM apps. This variation may be due to IM apps are more popular. The analysis for spreadsheet apps does not indicate the number of discarded apps in step 2. Step 3 (identify main functionalities) got 12 (52.17% from step 1) spreadsheet apps, 8 (3.46%) diabetes apps and we obtained 39 (16.04%) IM apps. This difference can be due to the increasing number of applications or to the main functionalities chosen (different for each type of application).

Step 5A (KLM analysis) revealed that, in average, the tasks in spreadsheet apps took between 2.1 and 4 interactions, in diabetes apps they took between 3.16 and 6.33 interactions and in IM apps they took between 4.2 and 7.8 interactions.

Regarding the MHE (Step 5B), the study on spreadsheet applications was not performed. In diabetes apps, the main usability issues detected were related to heuristics G and H, and related to IM apps were about heuristics B, C and G. These results on the heuristics suggest that the design is generally not good and it can be improved.

The methodology used has a number of advantages [4, 11], but it also has some disadvantages and limitations, for instance: some steps take a long time and can be tedious to perform, large number of elements in the initial stages, results are time sensitive, etc. Furthermore, detecting all usability issues in this kind of experiment is not possible because the context of use is not taken into account [3] [14] but, on the contrary, as a laboratory experiment there is more control over the usability issues detected. An important limitation of this study is that it was conducted only on the iOS platform. On other platforms, different results could be obtained.

IV. CONCLUSIONS AND FUTURE WORK

According to the results of the KLM, sending a message, replying to a message and adding contacts are usually the fastest functionalities to be completed. Moreover, deleting a contact or a chat usually becomes a serious problem. The applications with lower levels of interactions were (from lowest to highest) Surespot encrypted messenger, Hike Messenger, HushHushApp, Kik Messenger, Touch, Hiapp Messenger and WhatsApp Messenger.

Regarding the Mobile Heuristic Evaluation with mobile experts, almost all applications had usability problems in performing the primary tasks. WhatsApp Messenger and HushHushApp obtained the best usability ratings. On the other hand, hike Messenger, Kik Messenger and Touch were negatively evaluated and presented critical usability, but did it well in KLM. This suggests that it is necessary to perform both the KLM and Heuristic Evaluation methods because if the results were based only in KLM the applications chosen would have many usability problems.

After finishing the study, we can propose some recommendations to improve the usability of instant messaging apps in mobile devices. Firstly, based on the KLM results, we can suggest the following recommendations:

- Each task should not exceed more than 5 or 6 interactions. It was observed that more than 8 interactions cause confusion in performing a task.
- Specifying the ID of a contact (username, phone number or email) should be enough for adding a contact. Other options (extra data such as name, last name, location, etc.) should be optional.

The heuristic evaluation results led us to propose the following guidelines:

- The interface should be carefully designed to ensure that all elements of the app are properly displayed in any position.
- Do not tolerate unrecoverable errors. It is always better displaying an error message than an unexpected shutdown of the app.

The usability recommendations proposed are a valuable resource for mobile app developers because they will improve the usability of their IM apps in mobile devices, thus achieving more downloads and users of their apps. As a future work, a new analysis will be carried out on other existing mobile platforms (e.g., Android) to compare results. Finally, after that we are planning to develop a mobile instant messaging application meeting the recommendations proposed, which will solve the main usability problems identified in existing applications.

 ISBN: 978-1-61208-438-1

ACKNOWLEDGMENT

Authors want to acknowledge support from the University of Alcala (TIFYC and PMI research groups).

REFERENCES

[1] M. L. Smith, R. Spence, and A. T. Rashid, Mobile phones and expanding human capabilities. Information Technologies & International Development, 7(3), 2011, pp. 77-88.

[2] E. Bertini, et al., "Appropriating heuristic evaluation for mobile computing," International Journal of Mobile Human Computer Interaction (IJMHCI), 1(1), 2009, pp. 20-41.

[3] D. Zhang, and B. Adipat, "Challenges, methodologies, and issues in the usability testing of mobile applications," International Journal of Human-Computer Interaction, 18(3), 2005, pp. 293-308.

[4] C. Martin, D. Flood, and R. Harrison, "A Protocol for Evaluating Mobile Applications," Proceedings of the IADIS, 2011.

[5] E. Abdulin, "Using the keystroke-level model for designing user interface on middle-sized touch screens," in CHI'11 Extended Abstracts on Human Factors in Computing Systems. ACM. 2011, pp. 673-686.

[6] D. Kieras, Using the keystroke-level model to estimate execution times. University of Michigan, 2001.

[7] B. A. Nardi, S. Whittaker, and E. Bradner. "Interaction and outeraction: instant messaging in action," in Proceedings of the 2000 ACM conference on Computer supported cooperative work.. ACM. 2000, pp. 79-88.

[8] R. E. Grinter, and L. Palen. "Instant messaging in teen life," in Proceedings of the 2002 ACM conference on Computer supported cooperative work. ACM. 2002, pp. 21-30.

[9] C. Lewis, and B. Fabos, "Instant messaging, literacies, and social identities," Reading research quarterly, 40(4), 2005, pp. 470-501.

[10] B. S. Gerber, M. R. Stolley, A. L. Thompson, L. K. Sharp, M. L. Fitzgibbon, "Mobile phone text messaging to promote healthy behaviors and weight loss maintenance: a feasibility study," Health informatics journal, 15(1), 2009, pp. 17-25.

[11] C. Martin, et al., "A systematic evaluation of mobile applications for diabetes management," in Human-Computer Interaction–INTERACT 2011. Springer, 2011, pp. 466-469.

[12] E. Garcia,, C. Martin, A. Garcia, R.Harrison, D. Flood, "Systematic Analysis of Mobile Diabetes Management Applications on Different Platforms," Information Quality in e-Health, 2011, pp. 379-396.

[13] D. Flood,, R. Harrison, C. Martin, K. McDaid, "A systematic evaluation of mobile spreadsheet apps," in IADIS International Conference Interfaces and Human Computer Interaction. 2011.

[14] J. Bastien, "Usability testing: a review of some methodological and technical aspects of the method," International Journal of Medical Informatics, 79(4) , 2010, pp. e18-e23.

[15] W. Hwang, and G. Salvendy, "Number of people required for usability evaluation: the 10±2 rule," Communications of the ACM, 53(5), 2010, pp. 130-133.

[16] E. DIN, 9241-11. Ergonomic requirements for office work with visual display terminals (VDTs)–Part 11: Guidance on usability. International Organization for Standardization, 1998.

 ISBN: 978-1-61208-438-1

Multi-Criteria Test Case Prioritization Using Fuzzy Analytic Hierarchy Process

Sahar Tahvili*†, Mehrdad Saadatmand*, Markus Bohlin*

*Swedish Institute of Computer Science (SICS), SICS Swedish ICT Västerås AB, Sweden
†Mälardalen University, Västerås, Sweden
Email: {sahart, mehrdad, markus.bohlin}@sics.se

***Abstract*—One of the key challenges in software testing today is prioritizing and evaluating test cases. The decision of which test cases to design, select and execute first is of great importance, in particular considering that this needs to be done within tight resource constraints on time and budget. In practice, prioritized selection of test cases requires the evaluation of different test case criteria, and therefore, test case prioritization can be formulated as a multi-criteria decision making problem. As the number of decision criteria grows, application of a systematic decision making solution becomes a necessity. In this paper, we propose an approach for prioritized selection of test cases by using the Analytic Hierarchy Process (AHP) technique. To improve the practicality of the approach in real world scenarios, we apply AHP in a fuzzy environment so that criteria values can be specified using fuzzy variables when precise quantified values are not available. One of the advantages of the proposed decision making process is that the defined criteria with the biggest and most critical role in priority ranking of test cases is also identified. We have applied our approach on an example case in which several test cases for testing non-functional requirements in a systems are defined.**

Keywords–Software testing, Test case prioritization, MCDM, Fuzzy AHP, NFR, Fault detection.

I. INTRODUCTION

As the role of software systems in our daily life grows, it becomes more and more important to evaluate and guarantee the quality of such products and ensure that they correctly operate and provide their expected functionality. One way toward this goal is testing of the software product before releasing it to the customers. In simple terms, testing basically means execution of the software system and code with controlled input, in order to evaluate its quality and identify potential problems in it. Through testing, we can increase our confidence in the quality of the product.

In this context, it is reasonable to assume that the more tests that are performed out of a diverse and high-quality test suite, the greater should be our confidence in the product quality be. But in practice, there is usually limited resources (in terms of time, budget, personnel, etc.) available and allocated for testing activities.

Among different testing activities, our focus in this paper, is on the prioritization of test cases in a test suite, regardless of how they are created (i.e., manually implemented, automatically generated, or a mixed approach). Considering such resource limitations, it becomes very important that from all possible test cases that can be considered for a system, a good subset, which fits the available resources will be selected. Selection of appropriate test cases from a test suite can be done based on a number of different criteria. Considering that in practice and in industrial settings a test suite can consist of a large number of different test cases, it is necessary to apply a systematic approach for the selection of appropriate test cases (based on the identified criteria) from the set of all test cases existing in the test suite.

Generally, the term test case selection is used to refer to the techniques which aim to reduce the number of test cases that are executed. Test case prioritization techniques, on the other hand, are used to order test cases in a way that the most important ones, i.e., those which can lead to a higher and increased rate of fault detection are run earlier [1]–[3]. Test case prioritization can be particularly necessary in performing regression testing which is also an expensive testing process [2], [4] where (some) test cases are selected and executed several times.

In [5], we have introduced an approach for prioritization of test cases based on the result from model-based trade-off analysis of non-functional requirements. In that approach, by performing analysis on the model of non-functional requirements, parts of the system that can have more severe problems with respect to the satisfaction of such requirements are identified. This is done by calculating a deviation indicator value for non-functional requirements as part of the model analysis.

Then, assuming that a cost-effort value for each test case and a total cost-effort budget to perform testing activities are known, we prioritize test cases that target requirements with higher deviation indicator value (thus potentially more problematic parts of the system) while fitting the total cost-effort budget available (prioritization and selection).

In this paper, the test case prioritization part of our work in [5] is extended to enable prioritization decisions based on multiple criteria. To enable this, we apply the Analytic Hierarchy Process (AHP) technique. The importance of multi-criteria test case prioritization solutions is also recognized and emphasized in the literature, see for example [6].

Since in practice it is generally hard or sometimes impossible to provide precise values for different criteria [7] and properties of a test case such as fault detection probability, we apply AHP in a fuzzy environment so that users can specify criteria values in the form of fuzzy variables (e.g., high, low, etc.) and thus make the overall approach more practical and usable in real scenarios.

The term decision making in a fuzzy environment means a decision making process in which the goals and/or the constraints, but not necessarily the system under control, are fuzzy in nature. This means that the goals and/or the constraints constitute classes of alternatives whose boundaries are not sharply defined [8].

 ISBN: 978-1-61208-438-1

While in this paper a particular set of criteria are selected based on which prioritization decisions are made, the fuzzy AHP approach can be well used for any other set and number of criteria. It should also be noted that in this context, our test case prioritization approach in this paper, prioritizes and orders test cases based on various criteria and not necessarily and merely based on their early rate of fault detection, which provides thus a broader scope of prioritization than what is defined, for instance, in [1], [3] for test case prioritization.

Another important property of the approach is that the most critical decision criterion, i.e., the criterion which has the biggest role in the ranking of test cases, is also identified. The main contributions of the paper are thus the following:

- a novel multi-criteria test case prioritization method based on AHP,
- application of the method in a fuzzy environment to relax the need of having precise values for criteria,
- an illustration of the method using a case study on laptop customization, and
- a brief analysis and discussion of the results.

The remainder of the paper is structured as follows. In Section I-A, the related literature is reviewed. Section II gives the theoretical background for fuzzification and fuzzy multiple-criteria decision-making. In Section III, the proposed approach and suitable test-case properties are described. Section IV gives an example of applying fuzzy AHP to the testing of non-functional properties of a customized laptop computer system. Finally, Section V gives conclusions and recommendation for future research.

A. Related Work

In [3], [4] the initial problem has been assumed as a single criterion decision making problem. In [4] test cases have been prioritized by the rate of fault detection and the authors used a weighted average of the percentage of faults detected, which is not a direct measure of the rate of fault detection. In [3] the authors used a code-coverage based greedy algorithm for prioritizing the test cases. Since single criterion decision making is hard to cover a larger or more complex problem, we investigate multi-criteria prioritization in the present work. Techniques of multi-criteria decision making are in contrast able to cover a large number of criteria for several test cases. We further define the initial problem in a fuzzy environment by using the linguistic variables. The effect of the criteria on the test cases have been interpreted as a fuzzy set which allows the expression of imprecise properties.

II. Background & Preliminaries

Testing an embedded system is a costly activity in terms of time and budget consumption. Considering these two limiting factors in testing of a system and the aim of companies to reduce time-to-market for their products, only a certain number of test cases can be selected to execute. In this section, we introduce AHP as a suitable decision making technique and redefine this method in a fuzzy environment. The AHP applies a Decision Support System (DSS) model in selection of alternatives. DSS is a computer based information system which supports data mining, decision modelling and prioritization to solve structured and unstructured problems [9].

A. Fuzzification

Fuzzy truth represents membership in vaguely defined sets, and variables over these sets are called fuzzy variables. From a user perspective, fuzzy properties are often described using linguistic variables. This section outlines the process of transforming a linguistic value into a fuzzy value. For a full introduction to fuzzy mathematics [10].

Definition 1: A linguistic variable indicates a variable whose values are words or sentences in a natural or artificial language [11].

As an example, a fuzzy property such as *age* could be described using the values “young”, “fairly old”, and “middle-aged”.

Fuzzification consists of the process of transforming the linguistic variables to fuzzy sets [12]. We use grade of membership to associate a value, indicating the degree of truth, to each linguistic term. Some basic concepts of fuzzy logic which are relevant for this work is based on the definitions provided by Zadeh [13], Yun Shi [12], Yang [14] and Kerre [15], and are revisited here briefly.

Definition 2: A fuzzy set is a pair (A, m_A) where A is a set and $m_A : A \rightarrow [0, 1]$; for each $x \in A$, $m_A(x)$ is called the grade of membership of x in (A, m_A).

The grades of membership of 0 and 1 correspond to the two possibilities of truth and false in an ordinary set [13].

Let $x \in A$; then x is called fully included in the fuzzy set (A, m_A) if $m_A(x) = 1$ and is called not included if $m_A(x) = 0$. The set $\{x \in A \mid m_A(x) > 0\}$ is called the support of (A, m_A) and the set is called a kernel, where x is a fuzzy member if $0 < m_A(x) < 1$ (see [14]).

One-dimensional membership functions have different shapes such as triangular, trapezoidal or Gaussian shape. In this paper, we use five triangular-shaped membership functions, illustrated in Figure 1.

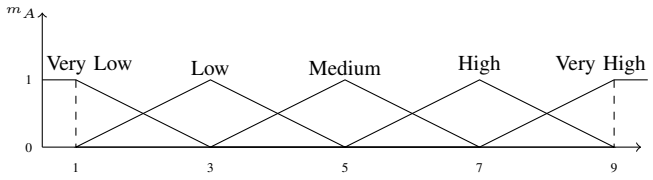


Figure 1. The five fuzzy membership functions for the linguistic variables

Definition 3: A triangular fuzzy number (TFN) can be defined as a triplet $M = (l, m, u)$ where l, m, u are real numbers and l indicates low bound, m is modal and u represents a high bound [16].

B. Fuzzy Multiple Criteria Decision Making

Multi-criteria decision making (MCDM) for structuring decision problems and evaluating alternatives provides a rich collection of methods [17]. The decision maker’s role in the decision making situations is to evaluate the effect of different criteria on the existing alternatives to choose the best one among them. In addition, the decision makers will need to choose an appropriate technique for decision making, which depends on the problem statement, limitation and constraints.

Since Zadeh and Bellman and a few years later Zimmermann developed the theory of decision support systems in a fuzzy environment, different techniques such as TOPSIS

 ISBN: 978-1-61208-438-1

(The Technique for Order of Preference by Similarity to Ideal Solution), QSPM (Quantitative Strategic Planning Matrix), AHP and etc., have been developed for solving various multi-criteria decision making problems [18].

In the present work, we use the fuzzy analytic hierarchy process (FAHP), for solving our problem. The AHP approach is an example of a heuristic algorithm, which based on the comparison matrix with triangular fuzzy numbers (TFN). Here, we provide a summary of how AHP is extended in a fuzzy environment.

As some other decision making techniques, AHP has also some weak points. One of the disadvantages of AHP is possible disagreement between decision makers. If, for example, more than one decision maker is working on the decision support system, different viewpoints about the linguistic variables of each criterion can complicate matters [19]. Therefore, using a TFN instead of a constant has been suggested as a good solution. By using Table I, the decision makers are able to interpret the linguistic variables in the form of TFNs.

TABLE I. THE FUZZY SCALE OF IMPORTANCE

Fuzzy number	Description	Triangular fuzzy scale	Domain	$m_A(x)$
$\tilde{9}$	Very High	$(7, 9, 9)$	$7 \leq x \leq 9$	$(x-7)/(9-7)$
$\tilde{7}$	High	$(5, 7, 9)$	$7 \leq x \leq 9$ $5 \leq x \leq 7$	$(9-x)/(9-7)$ $(x-5)/(7-5)$
$\tilde{5}$	Medium	$(3, 5, 7)$	$5 \leq x \leq 7$ $3 \leq x \leq 5$	$(7-x)/(7-5)$ $(x-3)/(5-3)$
$\tilde{3}$	Low	$(1, 3, 5)$	$3 \leq x \leq 5$ $1 \leq x \leq 3$	$(5-x)/(5-3)$ $(x-1)/(3-1)$
$\tilde{1}$	Very Low	$(1, 1, 3)$	$1 \leq x \leq 3$	$(3-x)/(3-1)$

As Table I represents, every linguistic variable has been defined by a TFN. In some disagreement situations, the geometric mean of the TFNs can be used as a final agreement. As mentioned earlier, AHP is based on a series of pairwise comparisons of alternatives and criteria.

In a fuzzy environment, the linguistic variables that we have defined in Table I based on the standard 9-unit scale [16], are used to make the pairwise comparisons. The fuzzy comparison matrix $A = (\tilde{a}_{ij})_{n\times n}$ can be formulated and structured as [20]:

$$A = \begin{pmatrix} (111) & \tilde{a}_{12} & \dots & \tilde{a}_{1n} \\ \tilde{a}_{21} & (111) & \dots & \tilde{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{a}_{n1} & \tilde{a}_{n2} & \dots & (111) \end{pmatrix} \quad (1)$$

where $\tilde{a}_{ij}$ $(i = 1, 2, ..., n, j = 1, 2, ..., m)$ is an element of the comparison matrix and the reciprocal property of the comparison matrix is defined as $\tilde{a}_{ij} = \tilde{a}_{ij}^{-1}$. The pairwise comparisons need to be applied on every criteria and alternatives, and the values for $\tilde{a}_{ij}$ come from a predefined set of fuzzy scale value as showed in Table I. Then $\tilde{a}_{ij}$ represents, a TFN in the form of $\tilde{a}_{ij} = (l_{ij}, m_{ij}, u_{ij})$ and matrix A consists of the following fuzzy numbers:

$$\tilde{a}_{ij} = \begin{cases} 1 & i = j \\ \tilde{1}, \tilde{3}, \tilde{5}, \tilde{7}, \tilde{9} \quad or \quad \tilde{1}^{-1}, \tilde{3}^{-1}, \tilde{5}^{-1}, \tilde{7}^{-1}, \tilde{9}^{-1} & i \neq j \end{cases}$$

After we create the comparison matrix A, we need to find a priority vector of matrix A.

To make it, we need to calculate the value of fuzzy synthetic extent $\tilde{S}_i$ for each row in matrix A by [16]:

$$\tilde{S}_i = \sum_{j=1}^{m} \tilde{a}_{ij} \otimes \left[\sum_{i=1}^{n} \sum_{j=1}^{m} \tilde{a}_{ij} \right]^{-1} \quad (2)$$

where $\tilde{a}_{ij}$ is a TFN, $\otimes$ is the fuzzy multiplication operator and:

$$\sum_{j=1}^{m} \tilde{a}_{ij} = \left(\sum_{j=1}^{m} l_{ij}, \sum_{j=1}^{m} m_{ij}, \sum_{j=1}^{m} u_{ij} \right), \forall i = 1, 2, ..., n, \quad (3)$$

also

$$\left[\sum_{i=1}^{n} \sum_{j=1}^{m} \tilde{a}_{ij} \right]^{-1} = \left(\frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{m} u_{ij}}, \frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{m} m_{ij}}, \frac{1}{\sum_{i=1}^{n} \sum_{j=1}^{m} l_{ij}} \right) \quad (4)$$

now we can compute the degree of possibility for the TFNs.

Definition 4: Let $\tilde{a}_1 = (l_1, m_1, u_1)$ and $\tilde{a}_2 = (l_2, m_2, u_2)$ be two TFNs, the degree of possibility of $\tilde{a}_1$ to $\tilde{a}_2$, $V(\tilde{a}_2 \geq \tilde{a}_1)$, can be obtained as [16]:

$$V(\tilde{a}_2 \geq \tilde{a}_1) = \begin{cases} 1 & \text{if } m_2 \geq m_1, \\ 0 & \text{if } l_1 \geq u_2, \\ \dfrac{l_1 - u_2}{m_2 - u_2 + m_1 - l_1} & \text{otherwise.} \end{cases} \quad (5)$$

then the degree of possibility for a convex fuzzy number can be calculated by:

$$V(\tilde{a}_2 \geq \tilde{a}_1) = \text{hgt}(\tilde{a}_1 \cap \tilde{a}_2) = \frac{l_1 - u_2}{m_2 - u_2 + m_1 - l_1} = d \quad (6)$$

where d is the ordinate of the highest intersection point between $\tilde{a}_1$ and $\tilde{a}_2$ (see Figure 2) and the term hgt indicates the height of fuzzy numbers on the intersection of $\tilde{a}_1$ and $\tilde{a}_2$ (see [16]).

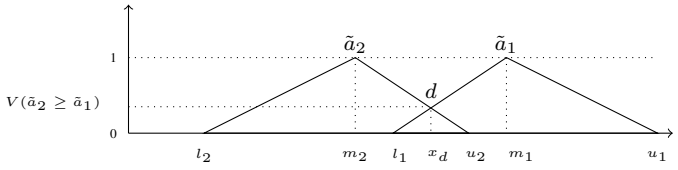


Figure 2. The degree of possibility for $\tilde{a}_2 \geq \tilde{a}_1$

Point x_d in Figure 2 indicates the point in the domain of $\tilde{a}_1$ and $\tilde{a}_2$ where the ordinate d is found [16]. Finally, we measure the weight vector for the criteria, assuming:

$$d'(A_i) = \min V(\tilde{S}_i \geq \tilde{S}_j), j = 1, 2, ..., n, j \neq i$$

where $A_i (i = 1, 2, \dots, m)$ are the m decision alternatives and n is the number of criteria, then the weight vector is obtained by [16]:

$$W'(A_i) = (d'(A_1), d'(A_2), ..., d'(A_m))^T, A_i (i = 1, 2, ..., m) \quad (7)$$

By normalizing Eq. (7) we are able to compute the

normalized weight vectors [21]:

$$W(A_i) = (d(A_1), d(A_2), ..., d(A_n))^T \quad (8)$$

where W is a non-fuzzy number and represents the arrangement of the alternatives.

Also by dividing the normalized weight for every criteria on the sum of the normalized weight vectors, we are able to compute the importance degrees of the criteria:

$$W_{C_j} = \frac{W(A_j)}{\sum_{i=1}^{n} W(A_i)}, j = 1, ..., n \quad (9)$$

where W_{C_j} represents the importance degrees of a criterion and n is the number of criteria.

In the next section through an example, we use FAHP by the mentioned equations for solving a multi criteria decision making problem.

III. Proposed Approach

In this section, we describe our approach on how to use FAHP to prioritize test cases. AHP serves as a powerful tool in calculating weights to solve a multi-criteria decision making problem, but this method is not able to handle uncertainty in the decision problems and also ranking of AHP is partly imprecise [22]. To solve decision making problems which consist of uncertainty and vagueness, fuzzy sets provides a pairwise comparison as an extension which provides a more accurate description of the linguistic variables within the process of decision making [22].

We summarize the steps of our approach for solving a typical DSS problem in the following points:

1) Criteria Identification
2) Alternative Determination
3) Effect Measurement
4) Fuzzification
5) Apply AHP Technique for DSS

In the first step of the proposed approach, we need to identify different criteria which have direct effect on the alternatives, a typical decision making problem can be defined as single or multiple criteria.

In the second step, we define and analyse some possible solutions which are referred to the alternatives.

In the third step, we measure the effect of the criteria on the every single alternatives, to perform this part, we can use linguistic or numerical variables, which depends on the decision making situation.

In the fuzzification part, we interpret our measurement into a fuzzy set by using fuzzy rules and reasoning. Note that the fuzzification part only applies to the linguistic variables, if we have some numerical or sharp values for the effect measurement part, we jump to the last step, which is applying a decision making technique to solve the initial problem.

As last step, we suggest AHP technique for DSS, other decision making technique such as TOPSIS, which covers the fuzzy rules, can be applied in this part.

In our previous work [5], we assumed the cost-effort of the test case as the only criterion in the decision making process. The cost-effort estimation for a test case could, for example, be the time, effort, and functional cost and budget that needs to be spent to perform each test.

In the present work, we assume a multiple-criteria decision making problem. The following set of test case properties are considered in this work as the main criteria for prioritization of test cases in the form of solving a DSS problem. The approach is not, however, limited to any particular set of test case properties as decision making criteria. In different systems and contexts users can have their own set of key test case properties based on which prioritization is performed.

- **Cost efficiency** (C_1) is used to capture the cost of a test case implementation, hardware setup cost, test case configuration cost (environment parameters), etc. The higher the cost efficiency degree of a test case, the more favorable it is.
- **Time efficiency** (C_2) is used to refer to a test case total execution time, test environment setup time and test case creation time. A test case with higher time efficiency is considered less time-costly.
- **Requirements coverage** (C_3) represents the number of requirements tested and covered by the test case.
- **Fault detection probability** (C_4) indicates the average probability of detecting a fault by each single test case.
- **Verdict Conclusiveness** (C_5) shows how conclusive and informative the verdict and result of a test case is. This is particularly interesting and more important for extra-functional aspects of a system where the meaning of a pass or fail result should be carefully investigated; for instance, a failure for user-friendliness or scalability [23].
- **Deviation Indicator** (C_6) is not per se and directly a property of a test case but that of a requirement. We have defined it in our previous work in [5] as the deviation degree of a requirement's satisfaction level from its ideal satisfaction level, i.e., when it is fully satisfied. It is calculated during the analysis of the requirements model. As it indicates which parts of a system can potentially have more severe problems with respect to the satisfaction of the requirements, we also include and use this property as one of the decision making criteria in the example case in this paper to also prioritize for test cases that target requirements with higher deviation indicator value.

IV. Case Scenario

In this section, through an example, we show how it becomes possible to use the result of model-based analysis to guide testing efforts. Hence, test cases can be prioritized by applying Fuzzy AHP. The application of the non-functional requirements (NFR) profile in building and customizing a laptop computer product, has been simulated by Figure 3. There are several non-functional requirements that are defined for this system such as low boot-up time, increased battery life and security and to satisfy each, several features are used and applied.

For example, to satisfy the security requirement, having the option to use the BIOS password checking at start up time and also finger print mechanism for authentication are considered. However, the use of such features has also impacts on other parts of the system. For example, using a password check during

 ISBN: 978-1-61208-438-1

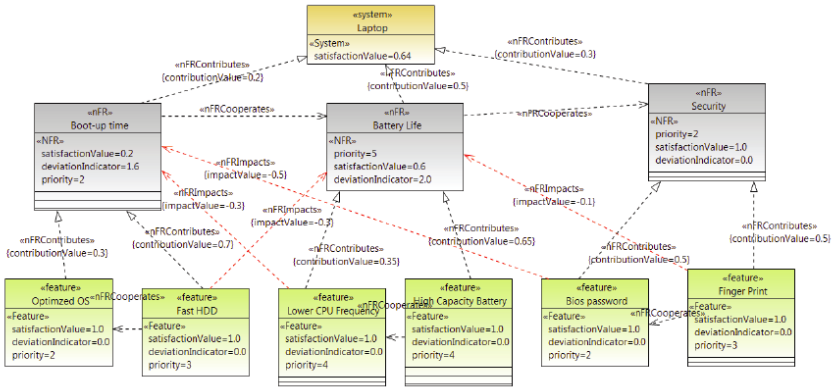


Figure 3. Analyzed model of the laptop system (Case Scenario)

the boot-up process affects the requirement to have low boot-up time negatively. Similarly, adding the finger print feature will add to the energy consumption of the system and thus affects the battery life time. These impacts and dependencies are established using the NFRImpacts links which are shown in red colour in Figure 3. The magnitudes of these impacts are stated on each of these links through the impactValue property. Customer preferences are captured by setting the priority properties. To test this system, there are 10 various test cases that target and cover its different requirements. As first step in the proposed approach, we need to identify the effective criteria and determine different alternatives.

Let $A = \{TC_1, TC_2, \ldots, TC_{10}\}$ be the set of test cases (alternatives) and $C = \{C_1, C_2, C_3, ..., C_6\}$ represents a set of the criteria that mentioned in proposed approach where $C_1 =$ Cost efficiency, $C_2 =$ Time efficiency, etc.

Figure 4 illustrates the relationships between the criteria and various test cases, as we see, the situation of decision making is symmetric where every criteria have a direct effect on every single test cases. The goal of this DSS problem is test cases prioritization. In the third step, we measure the effect of

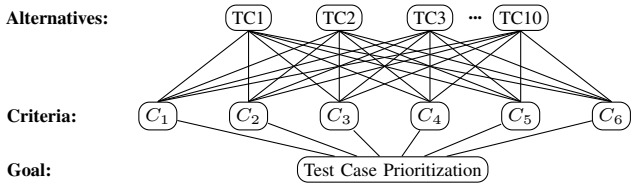


Figure 4. AHP hierarchy for prioritizing test cases

the criteria on the every single test cases. This effect has been assumed by the linguistic variables (e.g., low, high, etc.) and has been summarized in Table II.

TABLE II. THE PAIRWISE COMPARISON MATRIX FOR THE CRITERIA, WITH VALUES VERY LOW(VL), LOW (L), MEDIUM (M), HIGH (H) AND VERY HIGH (VH)

Test Case	Req. ID	C_1	C_2	C_3	C_4	C_5	C_6
TC1	RQ2	H	H	VH	H	VH	H
TC2	RQ1	M	M	H	M	M	VH
TC3	RQ1	M	H	H	H	VH	L
TC4	RQ3	VL	H	M	H	H	VH
TC5	RQ2	VH	M	M	VH	H	VH
TC6	RQ3	L	H	VH	H	VH	M
TC7	RQ1	M	L	L	VH	H	H
TC8	RQ3	VL	H	H	M	M	VH
TC9	RQ3	L	VH	VH	M	M	L
TC10	RQ2	VH	H	M	H	VH	M

The data in Table II are an empirical validation of the criteria effects. In the fuzzification phase (step 4), we interpret the effect of the various criteria on the test cases to a fuzzy set, to do these, Figure 1 and Table I have been used. As last step of the proposed approach, we apply AHP technique for prioritizing the test cases by using Eqs. (1) to (8).
The fuzzy pairwise comparison matrices for the alternatives and criteria becomes as follow by using Eq. (1):

$$C_1 = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_{10} \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_{10} \end{array} \\ \begin{pmatrix} \tilde{1} & \tilde{2} & \cdots & \tilde{2}^{-1} \\ \tilde{2}^{-1} & \tilde{1} & \cdots & \tilde{3} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{2} & \tilde{3}^{-1} & \cdots & \tilde{1} \end{pmatrix} \end{array} \quad C_2 = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_{10} \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_{10} \end{array} \\ \begin{pmatrix} \tilde{1} & \tilde{2}^{-1} & \cdots & \tilde{5} \\ \tilde{2} & \tilde{1} & \cdots & \tilde{3} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{5}^{-1} & \tilde{3}^{-1} & \cdots & \tilde{1} \end{pmatrix} \end{array}$$

$$C_3 = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_{10} \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_{10} \end{array} \\ \begin{pmatrix} \tilde{1} & \tilde{4} & \cdots & \tilde{1} \\ \tilde{4}^{-1} & \tilde{1} & \cdots & \tilde{1} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{1}^{-1} & \tilde{1}^{-1} & \cdots & \tilde{1} \end{pmatrix} \end{array} \quad C_4 = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_{10} \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_{10} \end{array} \\ \begin{pmatrix} \tilde{1} & \tilde{6} & \cdots & \tilde{2} \\ \tilde{6}^{-1} & \tilde{1} & \cdots & \tilde{3} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{2}^{-1} & \tilde{3}^{-1} & \cdots & \tilde{1} \end{pmatrix} \end{array}$$

$$C_5 = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_{10} \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_{10} \end{array} \\ \begin{pmatrix} \tilde{1} & \tilde{8} & \cdots & \tilde{1} \\ \tilde{8}^{-1} & \tilde{1} & \cdots & \tilde{2} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{6}^{-1} & \tilde{3}^{-1} & \cdots & \tilde{1} \end{pmatrix} \end{array} \quad C_6 = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_{10} \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_{10} \end{array} \\ \begin{pmatrix} \tilde{1} & \tilde{5} & \cdots & \tilde{9} \\ \tilde{6}^{-1} & \tilde{4}^{-1} & \cdots & \tilde{7} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{9}^{-1} & \tilde{8}^{-1} & \cdots & \tilde{1} \end{pmatrix} \end{array}$$

TFNs are calculated by using Eq. (3) as explained in the background section:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \tilde{a}_{ij} = (43.1, 72.4, 98.5)$$

and also Eq. (4) gives us the inverse of numbers:

$$\left[\sum_{i=1}^{n}\sum_{j=1}^{m} \tilde{a}_{ij}\right]^{-1} = (0.023, 0.013, 0.010)$$

the value of fuzzy synthetic extent $\tilde{S}_i$ can be obtained by using Eq. (2):

$$S_1 = (0.01, 0.09, 0.11), S_2 = (0.02, 0.0.24, 0.21)$$
$$S_3 = (0.01, 0.02, 0.03), S_4 = (0.06, 0.23, 0.24)$$
$$S_5 = (0.03, 0.33, 0.36), S_6 = (0.01, 0.15, 0.18)$$

we determine the weight rating for each criteria by using Eq. (5),

 ISBN: 978-1-61208-438-1

define $V_{ij} = V(\tilde{S}_i \geq \tilde{S}_j)$ then:

$$V_{12} = 1, V_{13} = 1, V_{14} = 1, V_{15} = 1, V_{16} = 1$$
$$V_{21} = 0.67, V_{23} = 1, V_{24} = 1, V_{25} = 1, V_{26} = 1$$
$$\vdots$$
$$V_{61} = 0.66, V_{62} = 0.48, V_{63} = 0.20, V_{64} = 0.90, V_{65} = 0.07$$

In this step, we compute the normalized and unnormalized weights for the criteria by using Eqs. (7), (8). The results have been summarized in Table III:

TABLE III. THE WEIGHT OF THE CRITERIA

Criteria	Unnormalized Weight	Normalized weight
C_1	0.58	0.02
C_2	0.67	0.25
C_3	0.28	0.01
C_4	0.97	0.36
C_5	0.02	0.22
C_6	0.07	0.10

In the next step we compare the weights for every single criteria with the alternatives, to avoid lengthy calculations we summarize the result in Table IV.

TABLE IV. COMPARISON THE WEIGHTS OF ALTERNATIVES WITH CRITERIA

Criteria / Weight	C_1 0.02	C_2 0.25	C_3 0.01	C_4 0.36	C_5 0.22	C_6 0.10
A_1	0.333	0.096	0.460	0.520	0.595	0.409
A_2	0.123	0.010	0.121	0.012	0.009	0.203
A_3	0.201	0.014	0.198	0.258	0.239	0.257
A_4	0.224	0.023	0.332	0.402	0.321	0.298
A_5	0.233	0.025	0.351	0.430	0.475	0.319
A_6	0.205	0.017	0.207	0.261	0.237	0.252
A_7	0.221	0.020	0.210	0.298	0.245	0.264
A_8	0.144	0.012	0.132	0.022	0.111	0.218
A_9	0.104	0.003	0.110	0.008	0.005	0.098
A_{10}	0.226	0.015	0.232	0.309	0.311	0.248

By normalizing the values in Table IV, we are able to prioritize the test cases, the result has been illustrated in Figure 5:

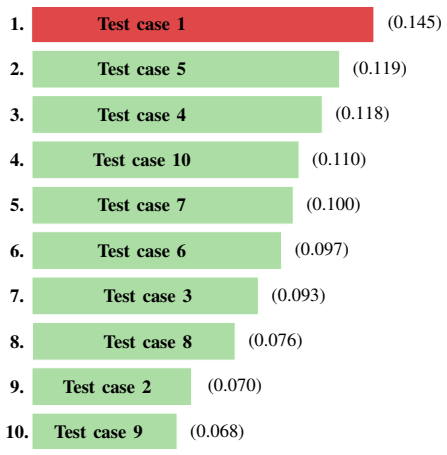


Figure 5. Test cases prioritization result

As can be seen, test case number 1 has the highest weight 0.145 among the test cases and it tests the requirement number 2. The proposed DSS approach prioritizes the test cases as the following set:

$$\{TC_1, TC_5, TC_4, TC_{10}, TC_7, TC_6, TC_3, TC_8, TC_2, TC_9\}.$$

Moreover, via Eq. (9), the importance degrees of the criteria are computed and summarized in Table V.

TABLE V. CRITERIA IMPORTANCE

Criteria	Importance
1. Fault Detection	0.37
2. Time Efficiency	0.26
3. Verdict Conclusiveness	0.22
4. Deviation Indicator	0.10
5. Cost Efficiency	0.02
6. Requirement Coverage	0.01

As can be seen, fault detection is identified as the most critical decision criterion which has the highest weight 0.37 among all the criteria.

As mentioned earlier, the most critical criterion is the one for which the smallest change in its current weight will alter the existing ranking of the alternatives [24].

V. Conclusion and Future Work

In this paper, we introduced a test case prioritization approach based on the fuzzy AHP decision making technique. Our approach enables to rank test cases based on a set of criteria and is particularly necessary when there is a limitation (due to resource constraints) in selection and execution of test cases for a system and it is not possible to run all the test cases. We demonstrated how the approach is applied using an example set of test case properties including cost efficiency, time efficiency, verdict conclusiveness, etc., that serve as criteria in the prioritization process. The approach is not, however, limited to this particular set of criteria. The extension and use of AHP in a fuzzy environment allows to specify the degree of a criterion in each alternative (i.e., a test case) using linguistic variables which relaxes the need of the users of the approach to specify precise quantified values. This can improve the practicality and adoption of the approach; for instance, where only estimated and imprecise data are available. As part of the decision making process, it was also identified which of the chosen criteria has a higher role in determining the ranking of the test cases. Use of this information on importance degree of each criteria in other analyses of test cases would be another interesting topic to further investigate.

As a future work, we are also going to examine other multi-criteria decision making techniques in prioritization of test cases. As another research direction of this work, finding a solution to apply a test case prioritization approach as part of the test case generation process would be an ideal scenario. Having such a solution would then enable to only generate a set of test cases which are analysed and deemed as feasible to execute with respect to the available resources for test execution. As for other works on test case prioritization techniques in the literature, generally fault detection rate is mainly used as the single important criterion in prioritizing and ranking of test cases. In this paper, we have considered test case prioritization in a broader aspect covering various and multiple criteria. Also in the application of the proposed approach, same priority level and importance was considered for all the criteria. If in a different context, one or more criteria are considered more important than the others (e.g., fault detection), a higher importance degree can be assigned to them during the pair-wise comparison process of the criteria. Considering the dependencies that can exist between test cases is also another extension of this work which we are currently investigating. Such dependency information

 ISBN: 978-1-61208-438-1

can as well be used in prioritization and ordering the execution of test cases. Therefore, there is the potential to combine and include it as another criterion in the decision making solution we proposed in this paper or use it in a separate step before or after the prioritization based on the other criteria.

VI. ACKNOWLEDGEMENTS

This work was supported by VINNOVA grant 2014-03397 through the IMPRINT project and the Swedish Knowledge Foundation (KKS) through the TOCSYC project. We would also like to thank Stig Larsson, Wasif Afzal and Daniel Sundmark for their constructive help and comments.

REFERENCES

[1] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in Proceedings of the 2007 International Symposium on Software Testing and Analysis, ser. ISSTA '07. New York, NY, USA: ACM, 2007, pp. 140–150.

[2] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "The effects of time constraints on test case prioritization: A series of controlled experiments," Software Engineering, IEEE Transactions on, vol. 36, no. 5, Sept 2010, pp. 593–617.

[3] A. Z. Dario Di Nucci, Annibale Panichella and A. D. Lucia, "Hypervolume-based search for test case prioritization," in Proceedings of the 2015 Symposium on Search-Based Software Engineering (SSBSE'15). Bergamo, Italy: Springer, 2015, pp. 140–150.

[4] G. Rothermel, R. Untch, C. Chu, and M. Harrold, "Test case prioritization: an empirical study," in Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on, 1999, pp. 179–188.

[5] M. Saadatmand and M. Sjödin, "On combining model-based analysis and testing," in Information Technology:New Generations (ITNG), 10th International Conference on, Las Vegas, USA, April 2013, pp. 260–266.

[6] M. Harman, "Making the case for morto: Multi objective regression test optimization," in Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, March 2011, pp. 111–114.

[7] T. L. Saaty, "How to make a decision:the analytic hierarchy process," European Journal of Operational Research, vol. 48, no. 1, 1990, pp. 9–26.

[8] R. Bellman and L. Zadeh., "Decision making in a fuzzy environment," Management Science, 1970, pp. 141–164.

[9] H. Jantan, A.Hamdan, and Z. Othman, "Intelligent techniques for decision support system in human resource management." 2010.

[10] V. Novák, I. Perfilieva, and J. Mockor, Mathematical principles of fuzzy logic. Springer Science & Business Media, 2012, vol. 517.

[11] D. Rabunal Dopico and Pazos, Encyclopedia of Artificial Intelligence. University of A Coruna, Spain, 2009, vol. 3.

[12] Y. Shi, "A Deep Study of Fuzzy Implications," Ph.D. dissertation, Ghent University, 2009.

[13] L. Zadeh, "Fuzzy sets," Information and Control, 1965, pp. 338–353.

[14] J. Yang and J. Watada, "Fuzzy clustering analysis of data mining: Application to an accident mining system," International Journal of Innovative Computing, Information and Control, 2012, pp. 5715–5724.

[15] E. E. K. B. De Baets, "Fuzzy relational compositions," Fuzzy Sets and Systems, vol. 60, no. 1, 1993, pp. 109 – 120.

[16] Y.-C. Tang, "Application of the fuzzy analytic hierarchy process to the lead-free equipment selection decision," Business and Systems Research, 2011, pp. 35–56.

[17] J. Malczewski, Multiple Criteria Decision Analysis and Geographic Information Systems, M. Ehrgott, J. R. Figueira, and S. Greco, Eds. Springer US, 2010, vol. 142.

[18] C. Carlsson and R. Fuller, "Fuzzy multiple criteria decision making: Recent developments," Fuzzy Sets and Systems, vol. 2, no. 1, June 1996, pp. 415–437.

[19] K. Shahroodi, S. Amini, E. Shiri, K. S. Haghighi, and M. Najibzadeh, "Application of analytical hierarchy process (ahp) technique to evaluate and selecting suppliers in an effective supply chain," Arabian Journal of Business and Management Review, vol. 1, no. 8, April 2012.

[20] T. Terano, Fuzzy Engineering Toward Human Friendly Systems. Ohmsha, 1992, no. v. 2.

[21] D.-Y. Chang, "Theory and methodology applications of the extent analysis method on fuzzy ahp," European Journal of Operational Research, 1996, pp. 649–655.

[22] G. Kabir and M. A. A. Hasin, "Comparative Analysis of AHP and Fuzzy AHP Models for Multicriteria Inventory Classification," International Journal of Fuzzy Logic Systems, vol. 1, no. 1, 2011, pp. 87 – 96.

[23] R. M. Hierons, "Verdict functions in testing with a fault domain or test hypotheses," ACM Trans. Softw. Eng. Methodol., vol. 18, no. 4, Jul. 2009, pp. 14:1–14:19.

[24] E. Triantaphyllou, B. Kovalerchuk, L. Mann, and G. M. Knapp, "Determining the most important criteria in maintenance decision making," Journal of Quality in Maintenance Engineering, vol. 3, no. 1, 1997, pp. 16–28.

 ISBN: 978-1-61208-438-1

Analysis of Optimization Requirement of Mobile Application Testing Procedure

Manish Kumar, Kapil Kant Kamal, Bharat Varyani, Meghana Kale
Centre for Development of Advanced Computing
Mumbai, India
email: kmanish@cdac.in, kapil@cdac.in, bharatv@cdac.in, meghanak@cdac.in

***Abstract*— The rapidly changing and demanding mobile application ecosystem has resulted in explosion of mobile applications among the users across the world. This behavior of the mobile application ecosystem inspires the mobile industries to make high availability of mobile application and development over the different platforms like Android, iPhone, Windows, BlackBerry, etc. The rapidly changing and demanding mobile application ecosystem has made mobile application development more complex and critical. A mobile application should be responsive, stable and secure. This high expectation with mobile application requires the right approach of testing. There are various mobile application testing strategies available, and in the current scenario of mobile application testing, we normally apply all the available testing strategies for all the mobile applications even which are not necessarily required. To optimize this bulky mobile application testing, we have to first analyze the mobile application and decide which testing strategies are required. In this paper, we have described various mobile application strategies available and have presented a classification of mobile applications based upon which mobile application testing procedure can be optimized.**

Keywords — Mobile Application Testing; Optimization; Graphic User Interface (GUI); SQLite.

I. INTRODUCTION

The rapidly changing and demanding mobile application development ecosystem can be understood by the fact that the number of mobile application download has increased by around 3000% in 2014 with approximately 138 billion downloads as compared to 4.5 billion downloads in 2010 [10]. The growing mobile application ecosystem has brought revolution in the mobile application development and it has made mobile application developers / providers to focus on creating mobile application testing strategies and road maps before releasing the mobile application for their users. It is important to build an application with all features and functionality required by the customer and which is beneficial to the application user, but it is even more critical to have a rigorous mobile testing plan before the mobile application is deployed or made available to its customer [1]. A comprehensive testing plan gives the confidence that the application will function as intended on different devices with varying screen sizes, resolutions, internal hardware, operating systems, and across any data transfer network. Today, there is a wide variety of mobile devices with different mobile operating systems, firmware updates and other possible customization that creates impossibly large sets of testing permutation and combination [8]. There is a large number of testing strategies available that makes mobile application testing more complex and a bulky procedure. Due to this complex process, extra use of resources increases the cost and time of complete testing. For optimization of mobile application testing, the proposed classification of mobile applications can help to broadly characterize the mobile applications first and then, to choose appropriate testing strategy.

The below mentioned goals can be achieved by using this classification to optimize the mobile application testing:

1. *Reduce the completion time of mobile application testing life cycle.*
2. *Reduce the testing cost and overall application cost.*
3. *Faster release of mobile application in market.*
4. *Reduce the process involved in mobile application testing with quality assured.*

In this paper, we describe how classification of mobile applications helps in optimizing the mobile application testing procedure. In Section II, we describe some of the available testing strategies in a mobile application testing life cycle. In Section III, we propose classification of mobile applications to optimize the mobile application testing procedure. In Section IV, we describe how the proposed classification helps in optimizing the mobile application testing procedure. In Section V, we present our conclusion.

II. AVAILABLE TESTING STRATEGIES

Once a mobile application is developed, it undergoes various testings before it is released. There is a large number of testing strategies available today. Some of the mobile application testings that are performed before a mobile application is released, are described below.

A. Installation/Uninstallation Testing

Installation is one of the important strategies of the testing activity. It is performed to verify if the software has been installed with all the necessary components and the application in the all targeted devices is working as expected. Installation would be the first user interaction with the end users so it should be perfect. To verify if installation testing is successful or not, we perform the following two-step check: 1. if application can be installed by following normal installation procedure; 2. if application is seen in installed applications list. To verify if uninstallation testing is successful or not, we check if all the components of the application are removed as expected during the uninstalling process [2].

B. User Interface Testing

User interface testing is required to examine how easy or user friendly the application is, to use for the real users. It is essential that a user interface is interactive and more relevant to task of mobile application. The user interface testing is

 ISBN: 978-1-61208-438-1

performed on different devices with different form factor to check if the screen (including text, images, etc) is displayed as intended. This test also includes text visibility in the selected language, navigation between screens and verification of functionality online / offline, feedback from interaction with the system, i.e., downloaded application should be prompt with message [3].

C. Functionality Testing:

The objective of functionality testing is to measure the quality of the functional components of the system. The functional testing verifies that the system behaves correctly from the user / business perspective and functions according to the requirements, models, storyboards, or any other design paradigm used to specify the application. The functional test determines if each component or business event: performs in accordance to the specifications, responds correctly to all conditions that may be presented by incoming events / data, moves data correctly from one business event to the next (including data stores), and that business events are initiated in the order required to meet the business objectives of the system [2].

D. Compatibility Testing

Compatibility test involves validating the application with different Mobile platforms, with different mobile devices, screen sizes, hardware and resolutions as per requirements. The compatibility test determines that the mobile application works exactly as we want it to, across all supported devices, platforms, screen sizes, OS versions.

E. Performance Testing

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing can be applied to understand the scalability of mobile applications. Performance testing includes the response time of application, application behaviour in change of different available networks, battery consumption and memory leaks, etc.

F. Security Testing

It includes the encryption/decryption techniques used for sensitive data communication, checks for multi users support without interfering with the data between them check for access to file saved in the app by any unintended users detect areas in tested application so that they do not receives any malicious content [3]. Also, the screen lock technique is such as Face Unlock, Voice Unlock, Pattern Unlock, Pin Unlock, Password Unlock, etc. also used to ensure security of mobile device.

G. Network Testing

It includes the testing of network availability, effect of speed of network on application and at not availability of network. We can check network testing in the scenario such as Signal loss, data loss over network, bandwidth, network delay, etc [5]. Unpredictable application behaviour, user-interface related errors, database corruption and functional issues are some of the impact on mobile application due to Network variability.

H. Service Testing

The Service Testing process is responsible for planning and coordinating tests to ensure that the specifications for the service design are met and validated through delivery of the service and, including co-operation with the Release and Deployment process, to manage and limit risks that could result from insufficient utility and warranty of the service in operation [6].

III. PROPOSED CLASSIFICATION OF MOBILE APPLICATION FOR OPTIMIZED TESTING STRATIGY

Systematic software engineering techniques should be employed to maximize the probability of finding faults with minimal resources, i.e., time and money. In the mobile application testing lifecycle as shown in Figure 1, the first step requirement/design analysis is very crucial and it needs to be done carefully. It is important that all the aspects of the mobile application are captured in the first step so that the further steps can be optimized.

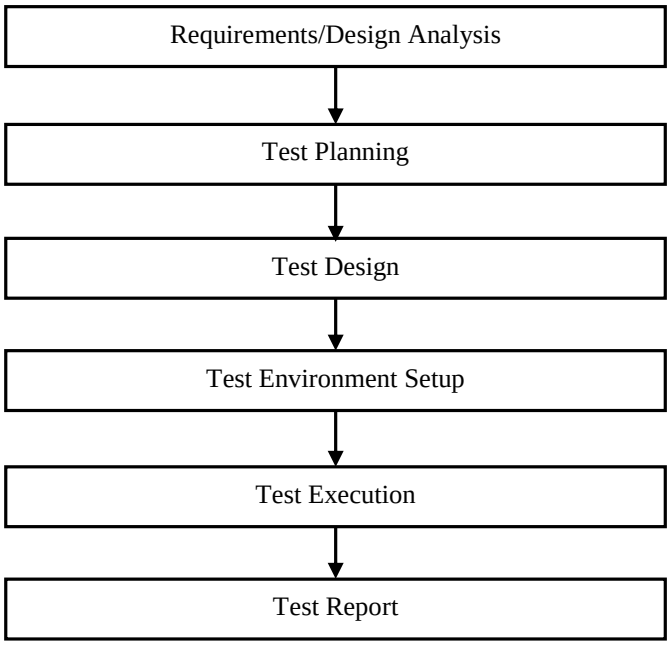


Figure 1. Mobile application testing Lifecycle

It is important to understand that mobile applications are different from traditional desktop applications because of the mobility, limited resources, context awareness, etc. [9]. Though the testing cycle may appear to be same for mobile application as compared to traditional application, the testing procedure for mobile application can be optimized if mobile application can be classified based upon various aspects. The mobile application runs on small screen size, limited resources (such as power, computational capacity, screen size, etc) as compared to a desktop. The mobile devices run on battery whereas desktops have continuous power supply. So for the optimization of mobile application testing, here we have presented mobile application classification and have analyze mobile applications on various aspects.

 ISBN: 978-1-61208-438-1

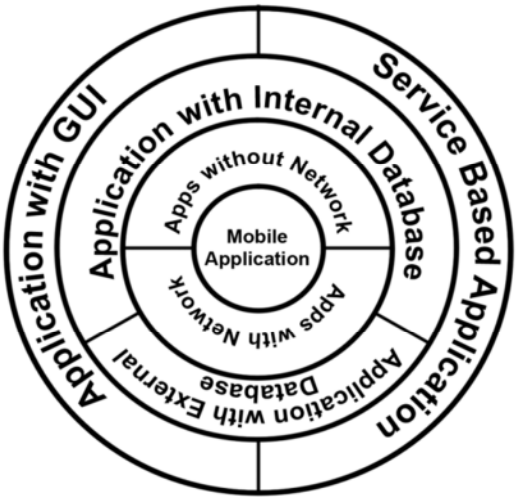


Figure 2. Proposed Classification of Mobile application

Figure 2 shows the proposed classifications of mobile application in perspective to optimize the mobile application testing.

A. Mobile application With GUI-With Internal Database-No Network[GIDNN]

It includes the mobile applications with Graphical User Interface (GUI) with internal database (database within device for example in Android SQLite) but no network requirement. This type of application does not require data network. For example, tic-tac game, which has the ability to store last scores and save the previous state of the game played by user in the past. Here, it is not required to test network and test the security of data communication channel because the data resides within the mobile device itself.

B. Mobile application With GUI-With Internal Database-With Network[GIDN]

It includes the mobile applications with GUI with internal database and also requires data network to submit the data to a remote server to get some required response. For example the application available on Mobile Seva AppStore named Ministry directory [4], initialized with its own internal database but when clicked on update it quickly repopulates its internal database with response data received from the update server. Another example is m-Indicator app on Google play store which stores the data in device database and also if we need exact location of express, rail alerts, etc. and then, it requires network connectivity.

C. Mobile application With GUI-With External Database-With Network[GEDN]

It includes mobile applications with GUI with external database and also requires data network to submit the data to remote server to store in external database. Mobile applications such as compliant registration or feedback for specific department need to store the data at department's end, so this type of application is GEDN. The mobile application of state government for birth registration, available on Mobile Seva AppStore [5], is another example of mobile application classified under GEDN, where the data filled by user in the GUI forms are sent over internet to state centralized server and the data is stored in centralized database.

D. Service based Mobile application With Internal Database-No Network[SIDNN]

These types of applications may or may not have any GUI. They start in background and run as background process. For example, in most of the smart phones there are various sensors which capture various data based upon various parameters. There are various internal applications which keep monitoring these sensors and send s information to other application which require information about these sensors. Internal database is used to store the data of past running.

E. Service based Mobile application With Internal Database-With Network[SIDN]

These types of mobile application may or may not have any GUI, they start in background, and the captured data is stored in internal database and may also communicate with remote servers. Example is Map applications, in this the latitude-longitude information is stored locally and location details are obtained from external GIS providers and the output is shown on map.

F. Service based Mobile application With External Database-With Network[SEDN]

These types of mobile applications do not have GUI, they capture data, send it to remote servers and the data is stored in the database of the remote server. Example of such application is version checking of mobile applications. If a new version or update is found at the remote application server, then the new version or the update is downloaded from the remote server and the mobile application is updated.

G. Service based Mobile application-With GUI-With Internal Database-With NoNetwork[SGIDNN]

These types of mobile applications have GUI to start or stop the application, and then they run as background service processes. For example the Contact application, which shows all contacts, recently dialed, missed call, received call, etc., which stores the contact info in the internal database, also shows time, duration and also there is no need of network in this type of applications.

H. Service based Mobile application-With GUI-With Internal Database-With Network[SGIDN]

These types of mobile applications have GUI to start or stop the service; this type of application, store data in internal database but require data network to communicate with the remote server or application. Example location based services; the maps application with Global Positioning System (GPS) will start your route mapping service then update it with its current locations and required data on map itself and also stores it in the internal database.

I. Service based Mobile application-With GUI-With External Database-With Network[SGEDN]

These types of mobile applications have GUI to start or stop the service; these type of applications do not store data internally and send the data collected to remote server. The data is stored in the external database. Example of this type of

 ISBN: 978-1-61208-438-1

mobile application is Drop Box application in which it has GUI from which we can see the data in our drop box and also we store new data on external database, i.e., on cloud directly through network connection.

IV. DISCUSSION

The paper presents a classification of mobile application based upon various aspects such as graphical user interface, data network requirement, database requirement, etc.. The mobile application testing can be optimized by using this classification and all the testing strategies may not be required for all the mobile application. Table 1, as shown below, describes various mobile application testing strategies which are required for different categories of mobile applications.

TABLE I. OPTIMIZATION TABLE FOR MOBILE APPLICATION TESTING USING PRPOSED CLASSIFICATION

TESTING STRATIGY / CLASSIFICATION	A	B	C	D	E	F	G	H
GIDNN	✓	✓	✓	✓	✓	x	x	x
GIDN	✓	✓	✓	✓	✓	✓	✓	x
GEDN	✓	✓	✓	✓	✓	✓	✓	x
SIDNN	x	✓	✓	✓	✓	x	x	✓
SIDN	x	✓	✓	✓	✓	✓	✓	✓
SEDN	x	✓	✓	✓	✓	✓	✓	✓
SGIDNN	✓	✓	✓	✓	✓	✓	x	✓
SGIDN	✓	✓	✓	✓	✓	✓	✓	✓
SGEDN	✓	✓	✓	✓	✓	✓	✓	✓

As shown in Table 1, it is not required to apply each and every testing approach for all the mobile applications but only those testing strategies which are required should be applied. For example, a mobile application classified under GIDNN category does not require service testing, security testing and network testing; so, these testings can be skipped from the complete cycle of testing process. This will reduce the testing time and cost. This in turn will reduce the overall cost of the mobile application and the reduced testing time helps in faster release of mobile application. There are some other aspects such as energy consumption, memory requirement, context awareness, etc., based upon which, the classification can be enhanced further.

V. CONCLUSION & FUTURE SCOPE

The paper presents a combined study of various mobile application testing strategies available and proposed classification of mobile application based upon various aspects such as graphical user interface, data network requirement, database requirement, etc.. The classification is simple yet the mobile application testing can be optimized by using this classification. This helps in bringing down the overall mobile application development cost and faster release of the mobile application. The proposed classification is not concrete and the classification can be further enhanced on other parameters. Testing strategies can also be defined specific to each classification to optimize the mobile application testing procedure further. The performance and reliability of the mobile application greatly depends upon the mobile device resources. We have not included the memory and energy requirements of mobile application which are crucial for performance and reliability testing.

ACKNOWLEDGEMENT

We are thankful to all the members of Mobile Seva team of C-DAC, Mumbai and Dr. Zia Saquib (Executive Director, C-DAC), Department of Electronics and Information Technology, Govt. of India for their direct as well as indirect contribution for this paper. We also thank the anonymous reviewers for their valuable insights and comments.

REFERENCES

[1] "Mobile Application Testing: Step by Step Approach", Retrieved from http://www.rapidvaluesolutions.com/ mobile-application-testing-step-by-step-approach/, accessed on 21 Sept 2015

[2] "Install / Uninstall Testing", Retrieved from http://www.tutorialspoint.com/software_testing_dictionary/pdf/install_uninstall_testing.pdf, accessed on 21 Sept 2015

[3] B. Kirubakaran and Dr. V. Karthikeyani, "Mobile application testing — Challenges and solution approach through automation", 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), Feb. 2013, pp. 79-84

[4] M. Kumar and M. Chauhan, "Best practices in Mobile application Testing", White Paper, Infosys

[5] "Mobile Seva Application store", https://apps.mgov.gov.in, accessed on 21 Sept 2015

[6] M. Jandial, A. Somasundara and Karthikeyan, "Enhance Mobile Application Performance with Successful Network Impact Testing", White Paper, Infosys

[7] M. Ryder, "Service Validation and Testing: A CA Service Management Process Map, Mar. 2010", Retrieved from http://acolyst.com/wp-content/uploads/2010/12/itil-service-validation-testing-tb-v2_234995.pdf, on 21 Sept 2015

[8] R. R. Nimbalkar, "Mobile Application Testing and Challenges", International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064, Vol. 2 Issue 7, July 2013, pp. 56-58

[9] H. Muccini, A. D. Francesco and P. Esposito, "Software Testing of Mobile Applications: Challenges and Future Research Directions", 7th International Workshop on Automation of Software Test (AST), June 2012, pp. 29-35

[10] "Number of mobile app downloads worldwide from 2009 to 2017", retrieved from http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/, on 30 Sept 2015

 ISBN: 978-1-61208-438-1

Property Based Verification of Evolving Petri Nets

Yasir Imtiaz Khan and Ehab Al-shaer

Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, USA
Email: ykhan2,alshaer@uncc.edu

***Abstract*—Software evolution is inevitable in the field of information and communication technology systems. Existing software systems continue to evolve to progressively reach important qualities such as completeness and correctness. Iterative refinements and incremental developments are considered to be well suitable for the development of evolving systems among other approaches. The problem with iterative refinements and incremental development is the lack of support of an adequate verification process. In general, all the proofs are redone after every evolution, which is very expensive in terms of cost and time. In this work, we propose a slicing based solution to add an adequate verification process to iterative refinements and incremental development technique. Our proposal has two objectives, the first is to perform verification only on those parts that may influence the property satisfaction by the analyzed model. The second is to classify the evolutions and properties to identify which evolutions require re-verification. We argue that for the class of evolutions that requires re-verification, instead of verifying the whole system only a part that is concerned by the property would be sufficient. We use Petri nets as a modeling formalism and model checking as a verification approach to show the viability of the proposed approach.**

Keywords–Software evolution; Re-verification; Model checking; Iterative refinements; Slicing.

I. INTRODUCTION

Software systems are playing an important role in our daily life. Companies are spending millions of dollars and are dependent on them. The software development process does not stop when a system is delivered, but continues throughout the lifetime of software. In general, existing software systems continue to evolve due to various reasons such as the emergence of new requirements, performance may need to be improved, business environment is changing [1] . According to the survey report conducted by Erlikh [2], 90% of software costs are software evolution costs and about 75% of all software professionals are involved in some form of evolution activity. These facts point out the importance of software evolution and demand tools and techniques for its better management.

Iterative refinements and incremental developments is a commonly used technique for handling complex systems in hardware and software engineering and is considered well suitable for software development and managing its evolution. The idea involves creating a new specification or implementation by modifying an existing one [3]. In general, the modeler provides a first model that satisfies a set of initial requirements. Then, the model can undergo several iterations or refinements until all the requirements are satisfied. In most cases, it is desirable for the developer to be able to assess the quality of model as it evolves.

The problem with the iterative and incremental development is that there is no guarantee that after each iteration or evolution of the model, it will still satisfy the previously satisfied properties.

Considering Petri nets as a modeling formalism and model checking as a verification technique all the proofs are redone which is very expensive in terms of cost and time. In this work, we propose a solution to improve the verification and re-verification of evolving systems by re-using, adapting and refining state of the art techniques. Our proposal pursues two main goals, the first is to perform verification only on those parts that may affect the property a model is analyzed for and the second is to classify evolutions, to identify which evolutions require re-verification. We argue that for a class of evolutions that require re-verification, instead of verifying the whole system only a part that is concerned with the property would be sufficient.

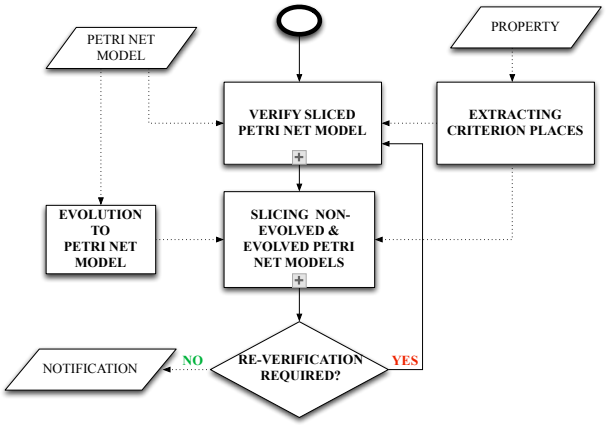


Figure 1. Process Flowchart property based verification of evolving Petri nets

Figure 1, gives an overview using Process Flowchart of the proposed approach, i.e., a slicing based verification of evolving Petri nets. At first, verification is performed on the sliced Petri net model by taking a property into an account. Secondly, we build slices for evolved and non-evolved Petri nets models. By comparing the resultant sliced models (i.e., Petri net and its evolved model), it is determined if the evolution has an impact on the property satisfaction and if it requires re-verification. In the worst case, if an evolution has an impact on the property

 ISBN: 978-1-61208-438-1

satisfaction only the resultant sliced evolved Petri net model would be used for the verification. The process can be iterated as per Petri net evolution.

The rest of the paper is structured as follows: in Section II, we give a informal and formal definition of Petri nets. In Section III, we give formal and informal description of the slicing algorithm and all the steps of slicing based verification of Petri nets. In Section IV, a slicing based solution is given for re-verification of evolving Petri nets. Details about the underlying theory and techniques are given for each activity of the process. In Section V, we discuss related work and a comparison with the existing approaches. In Section VI, we draw the conclusions and discuss future work concerning to the proposed work.

II. Informal and Formal Definition of Petri nets

Petri nets are a very well known formalism to model and analyze concurrent and distributed systems indroduced by C.A. Petri in his Ph.D. Dissertation [4].

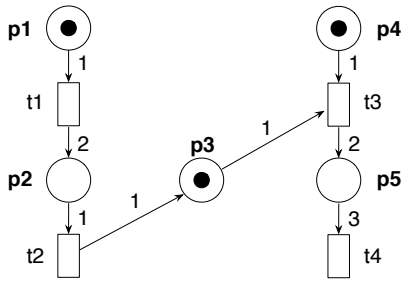


Figure 2. Example of a Petri net model

A Petri net is a directed bipartite graph, whose two essential elements are places and transitions. Informally, Petri nets places hold resources (also known as tokens) and transitions are linked to places by input and output arcs, which can be weighted. Usually, a Petri net has a graphical concrete syntax consisting of circles for places, boxes for transitions and arrows to connect the two. Formally, we can define :

Definition 1. Petri net: A Petri Net is: $PN = \langle P, T, w, m_0 \rangle$ consist of

$\circ$ P and T are finite and disjoint sets, called places and transitions, resp.,

$\circ$ a function $w : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$, assigns weights to the arcs,

$\circ$ a marking function $m_0 : P \rightarrow \mathbb{N}$.

The semantics of a Petri net expresses the *non-deterministic* firing of transitions in the net. Firing a transition means consuming tokens from a set of places linked to the input arcs of a transition and producing tokens into a set of places linked to the output arcs of a transition. A transition can be fired only if its incoming places have a token quantity greater or equal to the weight attached to the arc. As shown in Figure 2, transitions *t1 and t2* are enabled from the initial marking and *non-deterministically* any one of them can fire. Let us consider that if *t1* fires, the result of transition firing will remove a token from place *p1* and adds a token to place *p2*.

Definition 2. (Pre(resp.Post) set places(resp.transitions) of PN): Let $pn =< P, T, f, w, m_0 >$ be a Petri net, $p \in P$ a place then, preset and postset of p, noted $\bullet p$ and $p\bullet$, are defined as follows:

$\bullet p = \{t \in T / w(t, p) > 0\}$.

$p\bullet = \{t \in T / w(p, t) > 0\}$.

Analogously $\bullet t$ and $t\bullet$ are defined. We also note $\bullet P$ and $P\bullet$ representing pre(resp.post) set of transtions of all the places in set P. $\bullet T$ and $T\bullet$ are defined Analogously.

III. Abstract Slicing

Petri net slicing is a syntactic technique, which is used to reduce a PN model based on a given *criteria*. A *criteria* is a property for which the PN model is analyzed for. A sliced part is equal to only that part of a PN model that may affect the *criteria*. Considering a property over PN model, we are interested to define a syntactically smaller PN model that could be equivalent with respect to the satisfaction of the property of interest. To do so the slicing technique starts by identifying the places directly concerned by the property. Those places constitute the *slicing criterion*. The algorithm then, keeps all the transitions that create or consume tokens from the criterion places, plus all the pre-set places for those transitions. This step is iteratively repeated for the latter places, until reaching a fixed point. (It is important to note that the proposed slicing algorithms preserve certain specific properties as we intentionally do not capture all the behaviors to generate a smaller sliced net). Many algorithms are proposed for slicing Petri nets and their main objective is to generate reduced sliced net [5]–[11]. The first slicing algorithm to generate reduced sliced net was proposed by Astrid Rakow by introducing a notion of *reading and non-reading transitions*. Later, this idea was adapted by Khan et al in the context of Algebraic Petri nets (i.e., an advancement of Petri nets) [6], [10].

Informally, *reading transitions* do not change the marking of a place, meaning they consume and produce the same token. On the other hand, *non-reading transitions* change the markings of a place (see Figure 3), meaning they consume and produce different tokens. A reduced sliced net can be generated by discarding the *reading transitions* (as reading transitions do not impact the behavior of Petri net) and to include only *non-reading transitions*. Formally, we can define reading and non-reading transitions:

Definition 6. (Reading(resp.Non-reading) transitions of Petri nets): Let $t \in T$ be a transition in a PN. We call t a reading-transition iff its firing does not change the marking of any place $p \in ({}^\bullet t \cup t^\bullet)$, i.e., iff $\forall p \in ({}^\bullet t \cup t^\bullet), w(p, t) = w(t, p)$. Conversely, we call t a non-reading transition iff $w(p, t) \neq w(t, p)$.

We extend the slicing proposal of Rakow and Khan et al by introducing a new notion of *neutral transitions*. The abstract slicing algorithm preserves properties expressed in CTL^*_{-X} formulas, we refer the interested reader to [12] for the detailed proofs. Informally, a *neutral transition* consumes and produces the same token from its incoming place to an outgoing place. The cardinality of incoming (resp.) outgoing arcs of a neutral transition is strictly equal to one and the

 ISBN: 978-1-61208-438-1

cardinality of outgoing arcs from an incoming place of a neutral transition is equal to one as well. Another restriction is that the cardinality of outgoing arcs from the incoming place of a neutral transition is strictly equal to one and the reason is that we want to preserve all possible behaviors of the net. We may loose some behaviors when we merge incoming and outgoing places if we allow more outgoing arcs from the incoming place of a neutral transition. The idea is to use *reading transitions and neutral transitions* to generate smaller sliced net.

Definition 7. (Neutral transitions of Petri nets): Let $t \in T$ be a transition in a PN. We call t a neutral-transition iff it consumes token from a place $p \in^\bullet t$ and produce the same token to $p' \in t^\bullet$, i.e., $t \in T \wedge \exists p \exists p' / p \in^\bullet t \wedge p' \in t^\bullet \wedge |p^\bullet| = 1 \wedge |^\bullet t| = 1 \wedge |t^\bullet| = 1 \wedge w(t,p) = w(t,p')$.

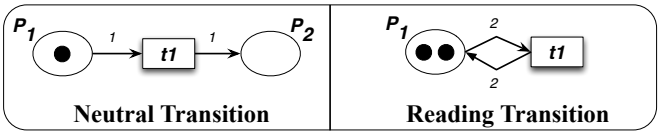


Figure 3. Neutral and Reading transitions of PN

1) Abstract Slicing Algorithm:: The abstract slicing algorithm starts with a Petri net model and a slicing criterion $Q \subseteq P$ containing place(s). We build a slice for an Petri net based on Q by applying the following algorithm:

Algorithm 1: Abstract slicing algorithm

AbsSlicing($\langle P, T, f, w, m_0 \rangle, Q$){
$T' \leftarrow \{t \in T / \exists p \in Q \wedge t \in (^\bullet p \cup p^\bullet) \wedge w(p,t) \neq w(t,p)\}$;
$P' \leftarrow Q \cup \{^\bullet T'\}$;
$P_{done} \leftarrow \emptyset$;
while $((\exists p \in (P' \setminus P_{done}))$ **do**
  while $(\exists t \in ((^\bullet p \cup p^\bullet) \setminus T') \wedge w(p,t) \neq w(t,p))$ **do**
    $P' \leftarrow P' \cup \{^\bullet t\}$;
    $T' \leftarrow T' \cup \{t\}$;
  end
  $P_{done} \leftarrow P_{done} \cup \{p\}$;
end
while $(\exists t \exists p \exists p' / t \in T' \wedge p \in^\bullet t \wedge p' \in t^\bullet \wedge |^\bullet t| = 1 \wedge |t^\bullet| = 1 \wedge |p^\bullet| = 1 \wedge p \notin Q \wedge p' \notin Q \wedge w(p,t) = w(t,p'))$ **do**
  $m(p') \leftarrow m(p') \cup m(p)$;
  $w(t,p') \leftarrow w(t,p') \cup w(t,p)$;
  while $(\exists t' \in^\bullet t / t' \in T')$ **do**
    $w(p',t) \leftarrow w(p',t) \cup w(p,t')$;
    $T' \leftarrow T' \setminus \{t \in T' / t \in p^\bullet \wedge t \in^\bullet p'\}$;
    $P' \leftarrow P' \setminus \{p\}$;
  end
end
return $\langle P', T', f_{|_{P',T'}}, w_{|_{P',T'}}, m_{0_{|_{P'}}} \rangle$;
}

In the Abstract slicing algorithm, initially T' (representing transitions set of the slice) contains a set of all the *pre and post* transitions of the given criterion places. Only the *non-reading transitions* are added to T'. P'(representing the places set of the slice) contains all the *preset* places of the transitions in T'. The algorithm then, iteratively adds other *preset* transitions together with their *preset* places in the T' and P'. Then, the *neutral transitions* are identified and their *pre and post* places are merged to one place together with their markings.

Considering an example Petri net model shown in figure 4, let us now apply our proposed algorithm on two example properties (i.e., one from the class of *safety* properties and one from *liveness* properties). Informally, we can define the properties:

ϕ_1 : *"The cardinality of tokens inside place P3 is always less than 5"*.

ϕ_2 : *"Eventually place P3 is not empty"*.

Formally, we can specify both properties in the *CTL* as:

$\phi_1 = \mathbf{AG}(|m(P3)| < 5)$.

$\phi_2 = \mathbf{AF}(|m(P3)| = 1)$.

For both properties, the slicing criterion $Q = \{P3\}$, since *P3* is the only place concerned by the properties. The resultant sliced Petri net can be observed in figure4, which is smaller than the original Petri net.

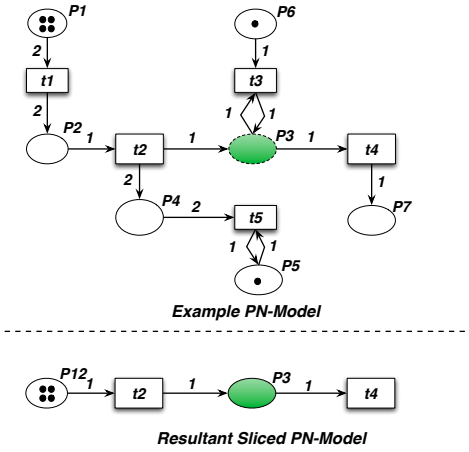


Figure 4. Petri net model and resultant sliced model after applying Abstract slicing algorithm

Let us compare the number of states required to verify the given properties without slicing and after applying abstract slicing. The total number of states required without slicing is ***985***, whereas with the sliced model number of states is ***15***.

IV. Classifications of Evolutions

The behavioral model of a system expressed in terms of Petri nets is subject to evolve, where an initial version goes through a series of evolutions generally aimed at improving its capabilities. Informally, Petri nets can evolve with respect to the structural changes such as: *add/remove places, transitions, arcs, tokens and terms over the arcs*. By notation, different Petri nets will be noted with superscripts such as $pn' = \langle P', T', f', w', m'_0 \rangle$. As there is no guarantee that after every evolution of a Petri net model, it still satisfies the previously satisfied properties. A naive solution is to repeat model checking after every evolution, which is very expensive in terms of time and space.

 ISBN: 978-1-61208-438-1

We propose a slicing based solution to improve the repeated model checking. Since it has already been proved that a sliced net is sufficient to verify properties. (Note: We refer the interested reader to [10], [12], [13] for the detailed proofs of all the theorems used in this paper and for slicing algorithms). According to our proposed approach, at first, slices

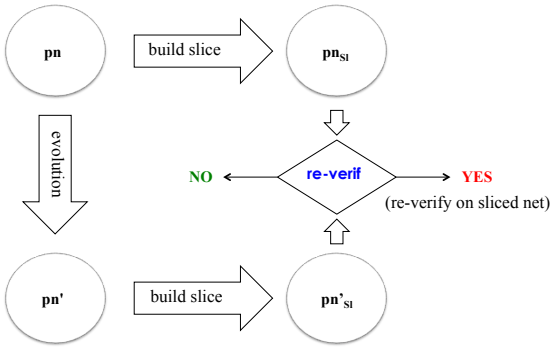


Figure 5. Overview

are generated for evolved and non-evolved Petri nets models with respect to the property by the abstract slicing algorithm as shown in Figure 5. Then, by comparing both sliced nets it is decided whether re-verification is required or not. If the answer is no then, re-verification is not required, whereas if the answer is yes, then, re-verification is performed on the sliced net. The good thing is that in both cases re-verification cost is improved. To decide for which evolutions re-verification is not required, we divide the evolutions into two major classes (by comparing both sliced Petri nets models as shown in the Figure 6), i.e., the evolutions that are taking place outside the slice, the evolutions that are taking place inside the slice. Furthermore, we divide the evolutions that are taking place inside the slice into two classes, i.e., the evolutions that disturb and those that do not disturb the previously satisfied properties.

A. *Evolutions taking place outside the Slice:*

The aim of slicing is to syntactically reduce a model in such a way that of the best reduced model contains only those parts that may influence the property the model is analyzed for. And if something is happening outside those parts of the system, then, it is guaranteed that previously satisfied properties are still true. We can generalize the notion, for all the evolutions that are taking place outside the slice do not influence the property satisfaction. Consequently, re-verification can be completely avoided for these evolutions. We formally specify how to avoid the verification if the evolutions are taking place outside the slice.

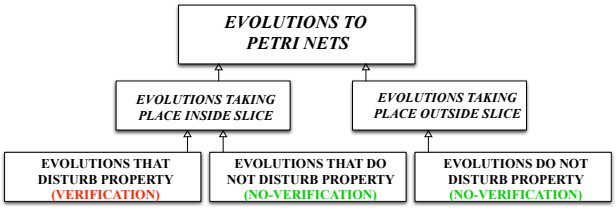


Figure 6. Classification of evolutions to Petri nets

Theorem 1: Let $pn_{sl} = \langle P, T, f, w, m_0 \rangle$ be a sliced Petri net model and $pn'_{sl} = \langle P', T', w', m'_0 \rangle$ be an evolved sliced Petri net model w.r.t the property ϕ. $pn_{sl} \models \phi \Leftrightarrow pn'_{sl} \models \phi$ if and only if

$$pn_{sl} = pn'_{sl}$$

Informally, this theorem states that if an evolution is taking place outside the slice then, the evolved Petri net model preserves the previously satisfied properties. According to the conditions imposed by the theorem, both the sliced net and evolved sliced net are same and if the Petri net model satisfy a given property then, this property will also be true in its evolved model. Conversely, if the Petri net model does not satisfy a given property then, this property will be false in its evolved model.

Let us recall the Petri net model and example property given in the section III. The example property is following $\mathbf{AG}(|m(P3)|) < 5)$. Figure 7, shows some possible examples of the evolutions to Petri nets model that are taking place outside the slice. All the places, transitions and arcs that constitute a slice with respect to the property are shown with the blue doted lines (remark that we follow the same convention for all examples). In the example evolution, weight attached to the arc between transition *t2* and place *P4* is changed and shown with the red color. For all such kind of evolutions that are taking place outside the slice, we do not require verification because they do not disturb any behavior that may impact the satisfaction of the property.

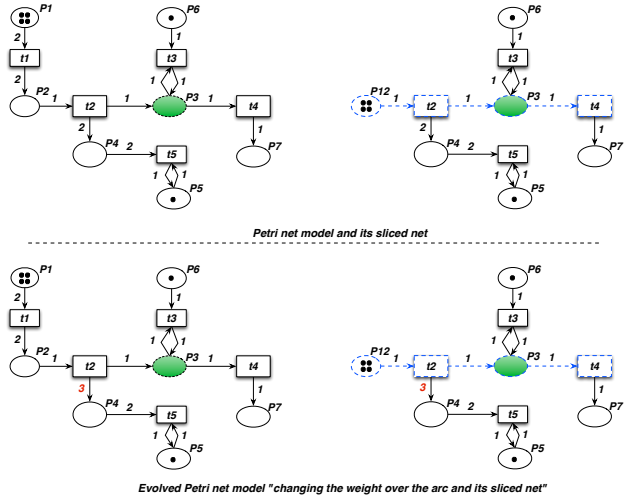


Figure 7. Evolutions to Petri net model taking place outside the slice

B. *Evolutions taking place inside the slice:*

For all the evolutions that are taking place inside the slice, we divide them into two classes, i.e., evolutions that require verification and the evolutions that do not require re-verification. Identifying such class of evolutions is extremely hard due to non-determinism of the possible evolutions. Specifically, in Petri nets small structural changes can impact the behavior of the model. It is also hard to determine whether a property would be disturbed after an evolution or it is still satisfied by the model.

 ISBN: 978-1-61208-438-1

To identify evolutions that are taking place inside the slice and do not require re-verification, we propose to use the temporal specification of properties to reason about the satisfaction of properties with respect to the specific evolutions. For an example, for all the safety properties specified by the temporal formula $\mathbf{AG}(\varphi)$ *or* $\exists\mathbf{G}(\varphi)$, if φ an atomic formula, using the ordering operators $\leq$ *or* $<$ between the *places* and their *cardinality* or tokens inside *places*, then, all the evolutions that decrease the tokens from places do not require re-verification because they do not impact the behavior required for the property satisfaction.

Theorem 2: Let $pn_{sl} = \langle P, T, f, w, m_0\rangle$ be a sliced Petri net and $pn'_{sl} = \langle P', T', w', m'_0\rangle$ be an evolved sliced Petri net model (in which tokens are decreased from places) w.r.t the property ϕ. For all the safety properties specified by temporal formulas, i.e., $\mathbf{AG}(\phi)$ *or* $\exists\mathbf{G}(\phi)$, and ϕ a formula using $\leq$ *or* $<$ ordering operator between the places and their cardinality or tokens inside places. $pn_{sl} \models \phi \Rightarrow pn'_{sl} \models \phi$ if and only if

$$\forall p \in (P \cap P')/m_0(p) \geq m'_0(p) \wedge T = T' \wedge f = f' \wedge w = w'$$

Let us recall the Petri net model and example property given in the Section III. The example property is following $\mathbf{AG}(|m(P3)| < 5)$, we can avoid the re-verification for several evolutions even if they are taking place inside the slice. Some possible examples of the evolutions are shown in Figure 8. In the first example, tokens are decreased from a place and in the second example, tokens are decreased from an arc, but the property is still satisfied.

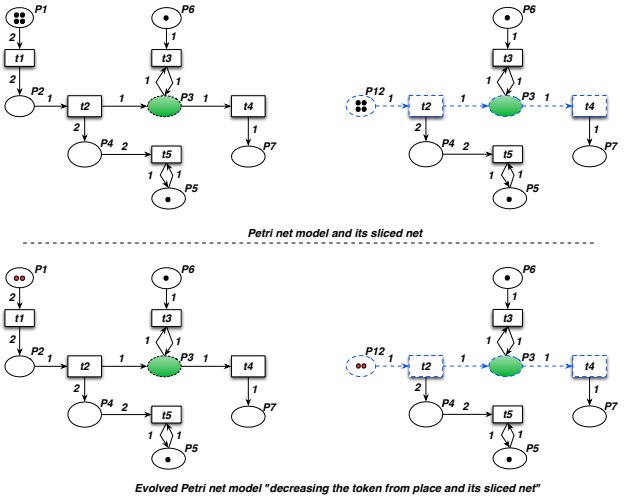


Figure 8. Evolutions to Petri net model taking place inside the slice

For all the liveness properties specified by a temporal formula $\exists\ \mathbf{F}(\varphi)$, and if φ a formula using the ordering operators ($\geq$ *or* $>$) the *places* and their *cardinality* or tokens inside *places* and their *values*, then, for all the evolutions that increase the token count, it is not required to verify them as they do not impact the behavior required for the property satisfaction.

Theorem 3: Let $pn_{sl} = \langle P, T, f, w, m_0\rangle$ be a sliced Petri net and $pn'_{sl} = \langle P', T', w', m'_0\rangle$ be an evolved sliced Petri net model (in which tokens are increased from places) w.r.t the property ϕ. For all the liveness properties specified by a temporal formula $\exists\mathbf{F}(\phi)$, and ϕ is using the ordering operators $\geq$ *or* $>$ between the *places* and their *cardinality* or tokens inside *places* and their *values*. $pn_{sl} \models \phi \Rightarrow pn'_{sl} \models \phi$ if and only if

$$\forall p \in (P \cap P')/m_0(p) \leq m'_0(p) \wedge T = T' \wedge f = f' \wedge w = w'$$

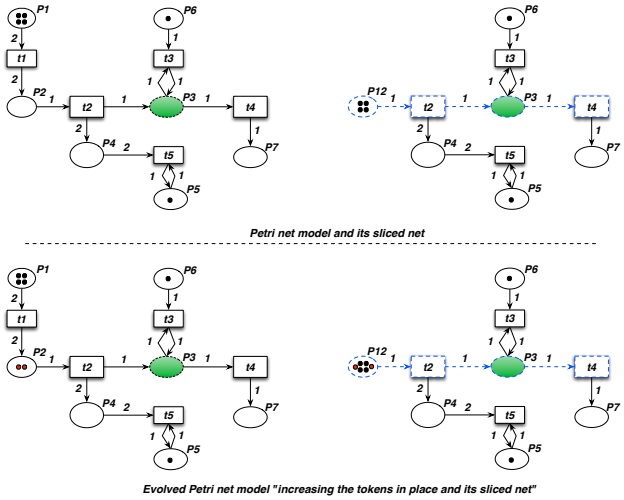


Figure 9. Evolutions to Petri net model taking place inside the slice

Let us consider again Petri net model given in the Section III , if we are interested to verify the example property such as: $\exists\mathbf{F}(|P3| > 3)$, verification can be avoided completely for several evolutions even if they are taking place inside the slice. Some possible examples of the evolutions are shown in Figure d9. In the first and second examples, tokens are increased but the property is still satisfied.

We identified above that for several specific evolutions and properties verification could be completely avoided, and for the rest of evolutions we can perform verification only on the part that concerns the property by following Section **??**. Even in this case we significantly improve the verification of evolution.

V. Related Work

Slicing is a technique used to reduce a model syntactically. The reduced model contains only those parts that may affect the property the model is analyzed for. Slicing Petri nets is gaining much attention in the recent years [5]–[11], [13]. Mark Weiser [14] introduced the slicing term, and presented slicing as a formalization of an abstraction technique that experienced programmers (unconsciously) use during debugging to minimize the program. The first algorithm about Petri net slicing was presented by Chang et al [5]. They proposed an algorithm on Petri nets testing that slices out all sets of paths, called concurrency sets, such that all paths within the same set should be executed concurrently. Astrid Rakow developed two slicing algorithms for Petri nets, i.e., CTL^*_{-X} slicing and *Safety slicing* in [10]. We introduced the Algebraic Petri net slicing for the first time [6], [12]. We adapt the notion of *reading and non-reading transitions* defined by Rakow [10] in the context of low-level Petri nets and applied to Algebraic Petri nets [6]. We extend the previous proposal by introducing

a new notion of *neutral transitions* and applied to Algebraic Petri nets [12]. In this work, we designed abstract slicing algorithm in the context of low-level Petri nets and used to reason about the re-verification. To the best of your knowledge this is the first proposal to use slicing to improve the re-verification of Petri nets models.

Most of the work regarding the improvement of the re-verification of evolving Petri nets is oriented towards the preservation of properties. Padberg and several other authors published extensively on the invariant preservation of APNs by building a full categorical framework for APNs, i.e., rule-based refinements [15]–[17]. Padberg consider the notion of a rule-based modification of Algebraic high level nets preserving the safety properties. The theory of a rule-based modification is an instance of the high-level replacement system. Rules describe which part of a net are to be deleted and which new parts are to be added. It preserves the safety properties by extending the rule-based modification of Algebraic Petri nets in contrast to transition preserving morphisms in [15]. These morphisms are called the place preserving morphisms by allowing transferring of specific temporal logic formulas expressing net properties from the source to the target net. Lucio presented a preliminary study on the invariant preservation of behavioral models expressed in Algebraic Petri nets in the context of an iterative modeling process [16]. They proposed to extend the property preserving morphisms in a way that it becomes possible to strengthen the guards without loosing previous behaviours.

In contrast to the property preservation, the scope of our work is broader. At first, we try to find out which evolutions require re-verification independent of the temporal representations of properties. Secondly, we focus on the specific properties and evolutions to improve the re-verification. We do not restrict the type of evolutions and properties to give more flexibility to a user. It is important to note that our proposed technique can further refine the previous proposals about the property preservation. The proposal is to preserve the morphisms restricted to the sliced part of the net.

VI. Conclusion and Future Work

In this work, we developed an approach to improve the verification and re-verification of systems modeled in Petri nets. At first, a Petri net model is syntactically reduced based on the given temporal property. The reduced model which we call a sliced model constitutes only that part of a model that may affect the property satisfaction. The sliced model preserves CTL^*_{-X} properties. Secondly, we classify evolutions and properties to determine whether re-verification is required or not. We do not restrict the types of evolutions and the properties to give more flexibility to the user. Our results show that slicing is helpful to alleviate the state space explosion problem of Petri nets model checking and the re-verification of evolving Petri nets.

The future work has two objectives; first is to implement the proposed approach. A tool named $SLAP_N$ (a tool for slicing Algebraic Petri nets) is under development [18]. It is important to note that the $SLAP_N$ tool is a generic tool over the Petri net classes such as Petri net, Algebraic Petri nets. It provides a graphical interface to draw a Petri net or Algebraic Petri net model together with the temporal description of properties. It contains the implementation of different slicing algorithms and a user can select any of them to generate a sliced Petri net model. The future work consists of implementation of the classification of evolutions and properties to automate the proposed approach. The second objective of future work is concerned to enhance the theory of preservation of properties. The aim is to develop a property preserving domain specific language for the evolving Petri nets based on the slicing and the classification of evolutions and properties proposed in this work.

References

[1] I. Sommerville, *Software Engineering: (Update) (8th Edition) (International Computer Science Series)*. Addison Wesley, June 2006.

[2] L. Erlikh, "Leveraging legacy system dollars for e-business," *IT Professional*, vol. 2, no. 3, pp. 17–23, May 2000.

[3] C. Larman and V. Basili, "Iterative and incremental developments. a brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.

[4] C. A. Petri, "Kommunikation mit automaten," Ph.D. dissertation, Universität Hamburg, 1962.

[5] J. Chang and D. J. Richardson, "Static and dynamic specification slicing," in *In Proceedings of the Fourth Irvine Software Symposium*, 1994.

[6] Y. I. Khan and M. Risoldi, "Optimizing algebraic petri net model checking by slicing," *International Workshop on Modeling and Business Environments (ModBE'13, associated with Petri Nets'13)*, 2013.

[7] M. Llorens, J. Oliver, J. Silva, S. Tamarit, and G. Vidal, "Dynamic slicing techniques for petri nets," *Electron. Notes Theor. Comput. Sci.*, vol. 223, pp. 153–165, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2008.12.037

[8] A. Rakow, "Slicing petri nets with an application to workflow verification," in *Proceedings of the 34th conference on Current trends in theory and practice of computer science*, ser. SOFSEM'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 436–447. [Online]. Available: http://dl.acm.org/citation.cfm?id=1785934.1785974

[9] ——, "Slicing and reduction techniques for model checking petri nets," Ph.D. dissertation, University of Oldenburg, 2011.

[10] ——, "Safety slicing petri nets," in *Application and Theory of Petri Nets*, ser. Lecture Notes in Computer Science, S. Haddad and L. Pomello, Eds., vol. 7347. Springer Berlin Heidelberg, 2012, pp. 268–287. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31131-4_15

[11] W. J. Lee, H. N. Kim, S. D. Cha, and Y. R. Kwon, "A slicing-based approach to enhance petri net reachability analysis," *Journal of Research Practices and Information Technology*, vol. 32, pp. 131–143, 2000.

[12] Y. I. Khan and N. Guelfi, "Slicing high-level petri nets," *International Workshop on Petri Nets and Software Engineering (PNSE'14) associated with Petri Nets'14)*, vol. 2, no. 3, pp. 201–220, 2014. [Online]. Available: http://ceur-ws.org/Vol-1160/

[13] Y. I. Khan, "Property based model checking of structurally evolving algebraic petri nets," Ph.D. dissertation, University of Luxembourg, 2015.

[14] M. Weiser, "Program slicing," in *Proceedings of the 5th international conference on Software engineering*, ser. ICSE '81. Piscataway, NJ, USA: IEEE Press, 1981, pp. 439–449.

[15] J. Padberg, M. Gajewsky, and C. Ermel, "Rule-based refinement of high-level nets preserving safety properties," in *Fundamental approaches to Software Engineering*. Springer Verlag, 1998, pp. 22 123–8.

[16] M. A. Q. Z. Levi Lucio, Eugene Syriani and H. Vangheluwe, "Invariant preservation in iterative modeling," *Proceedings of the ME 2012 workshop*, 2012.

[17] S. P. Er, "Invariant property preserving extensions of elementary petri nets," Technische Universitat Berlin, Tech. Rep., 1997.

[18] Y. I. Khan and N. Guelfi, "Slapn: A tool for slicing algebraic petri nets," *International Workshop on Petri Nets and Software Engineering (PNSE'14) associated with Petri Nets'14)*, vol. 2, no. 3, pp. 343–345, 2014.

 ISBN: 978-1-61208-438-1

Dynamic Evolution of Source Code Topics

Khaled Almustafa
College of Engineering
Prince Sultan University
Riyadh 11586, Saudi Arabia
Email: kalmustafa@psu.edu.sa

Mamdouh Alenezi
College of Computer and Information Sciences
Prince Sultan University
Riyadh 11586, Saudi Arabia
Email: malenezi@psu.edu.sa

***Abstract*—Open-source projects continue to evolve that result in so many versions. Analyzing the unstructured information in the source code is based on the idea that the unstructured information reveals, to some extent, the concepts of the problem domain of the software. This information adds a new layer of source code semantic information and captures the domain semantics of the software. Developers shift their focus on which topic they work more in each version. Topic models reveal topics from the corpus, which embody real world concepts by analyzing words that frequently co-occur. These topics have been found to be effective mechanisms for describing the major themes spanning a corpus. Previous Latent Dirichlet Allocation (LDA) based topic analysis tools can capture strengths evolution of various development topics over time or the content evolution of existing topics over time. Regrettably, none of the existing techniques can capture both strength and content evolution. In this work, we apply Dynamic Topic Models (DTM) to analyze the source code over a period of 10 different versions to capture both strength and content evolution simultaneously. We evaluate our approach by conducting a case study on a well-known open source software system, jEdit. The results show that our approach could capture not only how the strengths of various development topics change over time, but also how the content of each topic (i.e., words that form the topic) changes over time which shows that our approach can provide a more complete and valuable view of software evolution.**

Keywords–Open source; Source code; LDA; Topic extraction; Software evolution.

I. INTRODUCTION

Program comprehension is an essential activity in the course of software maintenance and evolution [1]. Typically, developers spend around 60% of their working hours comprehending the system while doing software maintenance tasks [1], especially the source code. Monitoring, visualizing and understanding the evolution of a large system are essentially challenging tasks.

Comprehending how source code topics evolve over time can be a great help for project stakeholders and managers to observe and understand activities and efforts performed on a software repositories over time. For instance, project managers can observe what feature the development team is working on by consulting the source code repository and developers can observe a specific feature evolution by consulting the same source code repository as well [2]–[4].

Several LDA-based techniques were proposed with the aim of supporting software projects stakeholders, managers, and developers to comprehend software evolution. Thomas et al. [4] used the Hall model [5] to study the history of source code to find out the strengths (i.e., popularity) of the change of several topics over time. They applied LDA one time on all versions of a specific software project to extract the topics and computed different metrics to show the strength of each topic for each version. Their approach can capture the development evolution strength. However, the content of a topic (i.e., the set of words that form a topic), never changes across the versions. Differently, Hindle et al. [6] used the Link model [7] that runs a separate LDA for each time window and used a different step to link similar topics. Their approach can capture changes in the content of each topic over time (i.e., content evolution). Unfortunately, their approach was not able to recover the strength of a topic across all time windows because of the lack of available information (time windows). Consequently, none of the current approaches can capture both strength and content evolution.

As we have seen, current approaches on understanding source code topics evolution emphasized on the strength of the evolution or the content evolution. However, both strength and content of the evolution are essential for developers to entirely comprehend how software evolves. For instance, project managers want to figure out how much effort is dedicated to feature X at a specific time, which can be attained by calculating topic strength. They may also want to figure out what kind of effort was done on feature X at a specific time point. By itself, topic strength will not assist project managers with this kind of information. Instead, the content of a specific topic can give some insights and shed some light on activities which performed on feature X at a specific point of time.

In this work, we propose a new approach to capture source code evolution from two different dimensions: strength and content. We apply Dynamic Topic Models (DTM) [8] on the source code of a software repository. After that, we capture topic strength evolution by calculating the Normalized-Assignment metric at each software release to represent the strength of a topic for that time. We capture topic content evolution by extracting the top 10 words that characterize a topic for each software release.

We conduct an empirical study on the source code of a well-known open source software system, jEdit. The results show that the new approach can capture both strength and contents of different source code topics over time, which corresponds to meaningful description of the whole development iteration, hence, a more complete view of software evolution.

An essential step towards comprehending and understanding the functional behavior of any system is to find and recognize business topics that exist in the source code of

 ISBN: 978-1-61208-438-1

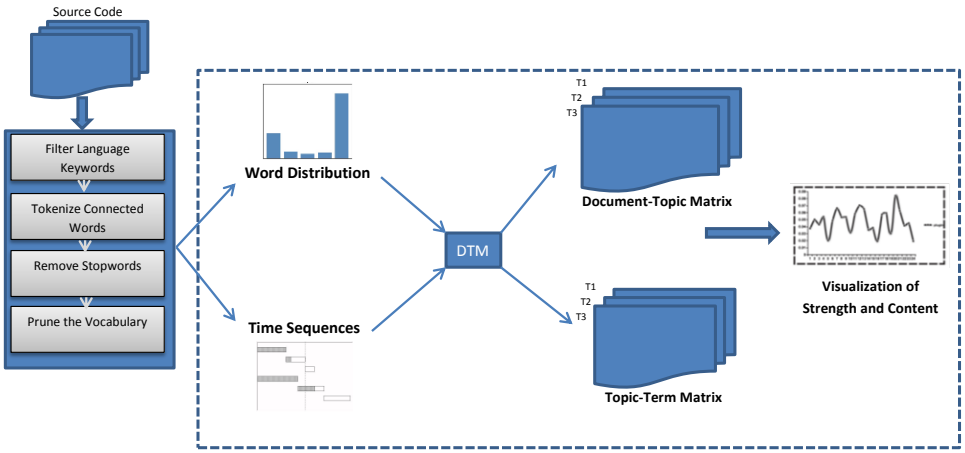


Figure 1. The Proposed Approach

the system. These business domain objects are modeled as high level components and then realized in the implementation and transformed into code. For example consider an UML modeling application that models UML diagrams and deals with objects, figures, relationships, and cardinality. When a maintainer with no application knowledge wants to add a new feature or modify one of the features, he/she will find it very difficult before comprehending and understanding the main functionality of the application. Extracting business topics from the source code and establishing the relationship between them would be a huge support in finding related data structures, methods, classes. This will eventually help the developer or the maintainer productivity, especially when dealing with a large system with little documentation.

The rest of the paper is organized as follows: Section II discusses topic evolution models in more depth. Section III discusses the approach used in this work. Section IV presents the experimental evaluation and discussions. Section V discusses threats to the validity of the study. Section VI discusses some related work to this work. Section VII concludes the paper.

II. Topic Evolution Models

Topic evolution model is taking into consideration the time while modeling the topics. It models how certain topics evolve and change over time. For a specific topic, the strength of that topic can change several times over time (spikes and drops) during the lifetime of a corpus. Furthermore, a specific topic content may likewise change over time, designating different aspects of the evolution within a specific topic. As we have seen in the introduction section, numerous topic evolution models were proposed in the literature.

1) The Link Model, which was proposed by Mei et al. [7]: Hindle et al. [6] were the first ones who applied this approach to analyze the topic evolution of commit messages. They utilized topic modeling (LDA) separately for each time. Then they used a post-processing phase to link similar topics across successive time interval.
2) The Hall model, which was proposed by Hall et al. [5]: Linstead et al. [9] were the first ones who applied this approach to analyze source code evolution. The same approach was used and validated by Thomas et al. [10] to model source code changes. The Hall model applies LDA to the whole versions of the software corpus (all releases included in the study). Then a post-processing phase is used to separate corpus at different versions and different metrics are calculated to represent how much the contribution of this topic in a specific version.
3) Dynamic Topic Models (DTM), which was proposed by Blei et al. [8]: DTM is what is used in this work. It models a topic evolution as a discrete Markov process with normally distributed changes between time periods that allows only steady changes over time. It uses time-sequentially organized documents of the corpus to capture the topics evolution and creates a document-topic matrix and topic-term matrix at each time period. Document-topic matrix represents each document as multi-membership mixture of topics to show the topic strength evolution. Topic-term matrix represents each topic as a multi-membership mixture of terms to show the topic content evolution

III. The Proposed Approach

Figure 1 shows a high-level overview of our approach. We first obtain the source code of the studied systems. Second, we filter out noisy data by applying a number of preprocessing steps on the corpus. Third, we determine the optimal number of topic for this corpus. Fourth, we obtain word distributions and the releases into time sequences. These distributions and sequences are used as input for DTM to produce document-topic matrix and topic-term matrix at each release. Our approach is very similar approach to Hu et al. [11] but it is different than their approach in two main aspects: the target of the topic modeling (commits vs source code) and choosing the number of topics (random vs well-established method).

A. Data Preprocessing

A number of pre-processing steps are applied to the source code [4]. These pre-processing steps are common in most

 ISBN: 978-1-61208-438-1

information retrieval techniques [1]. First, syntax and programming language keywords are filtered out. Second, each word is then tokenized according to well-known naming practices, for instance, underscores (first_name) and camel case (firstName). Third, common English terms are removed (stop words) to eliminate noise. The final step is to prune the vocabulary. The number of terms that can end up the bag-of-words is very large, which usually would cause a problem in most text-mining applications. In order to select the most useful subset, a filter has been applied to remove the overly common terms that appear in too many documents (=90%), as they can be seen as a non-informative and background terms.

B. Choosing K

Choosing how many topics to use in topic models is still a research problem not only for the source code domain. Topic excerption in textual documents also encounters the same problem. In this work, we adopted a well-known method for determining the number of topics [12]. This method specifies that the number of topics K can be determined by running LDA for different values of K with freezing the LDA hyper-parameters. For each value of K, they estimate the symmetric Kullback-Leibler divergence [13] between the singular values of the topic-word matrix and the document-topic matrix using the following equation:

$$Measure = KL(CM1||CM2) + KL(CM2||CM1) \quad (1)$$

The method calculates the symmetric Kullback-Leiber divergenece of the Singular value distributions of of two matrices M1 and M2. In this equation, CM1 represents the singular values distribution of the topic word matrix, CM2 represents the distribution obtained by normalizing the vector L*M2 where L is a one-dimensional vector of documents lengths in the corpus and M2 is the document topic-matrix. To determine K, choose K where the minimum value of the measure is. It is noteworthy that these distributions CM1 and CM2 are in sorted order.

C. Running Dynamic Topic Models

After we filtered noisy words in the corpus by applying the pre-processing steps, we use DTM to generate the document-topic matrix and topic-term matrix for each release after we feed the DTM the word distribution and the time sequences. Dynamic topic modeling applies these steps using several sequential time slices in the data set. We wrote an R script that applies our approach. We used the 'topicmodel' package version 0.2-1 in the R language version 3.1.12. The LDA parameters were chosen based on the recommendation of the literature [14]. The used parameters are = 50/K and = 0.01 where K is the number of topics.

D. Visualization of Strength and Content

After applying our approach, we visualize the results using the document-topic matrix and topic-term matrix. We calculate several topic metrics to symbolize both the strength and content of topics evolutions. We measure how the topic strength changes over time by computing a normalized assignment metric at each time point. This metric is the average value of the topic memberships of all documents in that topic at a time, which indicates the total presence of the topic in that time. The strength evolution of a topic is a time-indexed vector of normalized assignment values for that topic. Then, for the topic content, which includes the words and their distributions, it contains two parts: the word and its frequency within a topic. We choose the top 10 most frequent words to illustrate a topic and measure how the topic content changes over time by computing Term-Frequency (TF) at each time point.

IV. RESULTS

In this section, we applied our approach to the repository of jEdit. We show the example topics and the captured information of version 3 of jEdit in Table I. We discuss our results and the empirical data in detail. The results are demonstrated in Figure 2, Figure 3, and Figure 4. These figures display a number of top words from selected topics in each version based on the term frequency (TF) value of each word in that topic. By looking at the figures, we observe the topic strength evolution by mapping the assignment value of topics to versions as well as the topic content evolution by mapping TF values of several top words to versions.

We applied our approach to the repository of jEdit from 2002 to 2012, which includes 10 main releases (3.0, 3.1, 3.2, 4.0, 4.1, 4.2, 4.3, 4.4.1, 4.5, 5.0). We found that most topics' strength evolution fluctuated greatly during the studied time period, which indicates that development topics are varied and distributed in each version. Furthermore, we observed that the top two words across different topics are add and plugin, which represents the active growth of plugins, an essential feature of jEdit. We studied each topics strength and content. We found three important topics:

Selection. Topic 1 relates heavily to selections done in jEdit. We found that the most frequent words contained in this topic were line, text, selection, length and area. The topic's strength reaches to peak (8%) in version 9 (4.5), and before that time it varies from 3.1% to 5.8% and fluctuates greatly. We looked in more details at the top words in version 9, and found that the TF of these words seems to decline after version 9. This was the real content evolution trend of this topic, because we found that the top words rarely appeared in previous versions.

Formatting. In topic 5, words such as color, window, font, height and width are common throughout the whole selected versions, which clearly represent the continuing availability of these features in jEdit. The topics strength varies from 1.9% to 8% and also experiences great fluctuation. The peak strength was reached in version 2, but after other functionalities, which implies other development topics, the strength fell down in consecutive versions.

GUI. In topic 7, words such as border, handler, action, button, panel, area and event are common during the whole time, meaning that several components relate to GUI. The topic's strength reaches to peak (about 15%) in version 8 and we found that about 20% of the top matching words come from this version. We also found that border is a notable GUI feature provided by jEdit because the word border is almost the top frequent term within this topic from the beginning to end.

There are other development topics, such as bug fixing and feature requests, directory for plugins, buffer and bufferset, regular expressions, build system files and code cleanup.

 ISBN: 978-1-61208-438-1

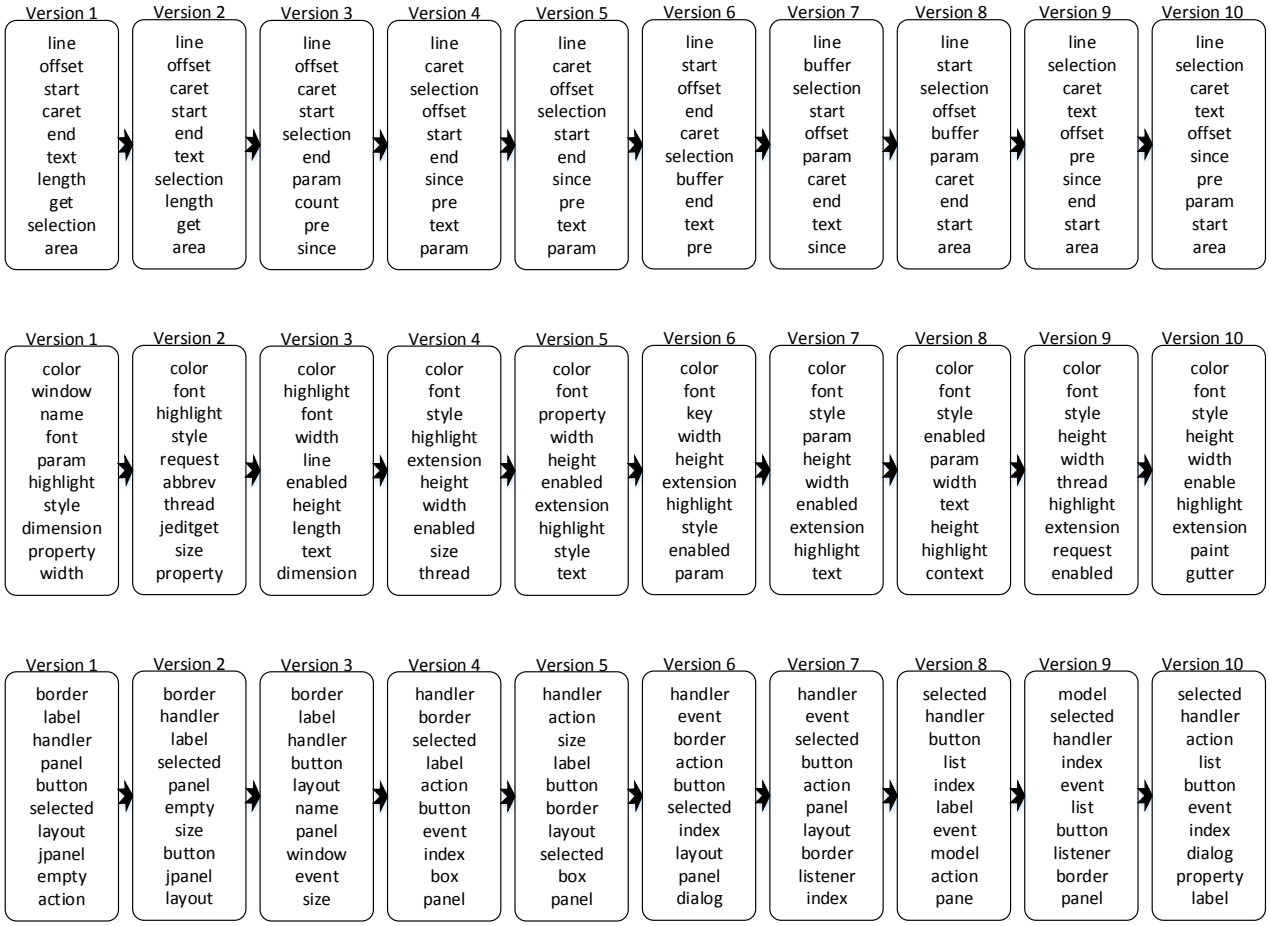


Figure 2. 10 Top Words Evolution for Topics.

TABLE I. EXAMPLE TOPICS AND THE CAPTURED INFORMATION OF VERSION 3 OF JEDIT

Topic	Top 10 Words
Selection	line, offset, caret, start, selection, end, param, count, pre, since
Formatting	color, highlight, font, width, line, enabled, height, length, text, dimension
Menu	border, label, handler, button, layout, name, panel, window, event, size

V. THREATS TO VALIDITY

In this section, we discourse some limitations to our study. The approach of this work depends on the quality of comments and identifier names found in the code. jEdit is known for its robust designs, extensive documentation, strict coding and naming conventions. In addition, a previous study revealed that majority of java systems have good comments and good identifiers names, which make them sufficient for such topic analyses [15].

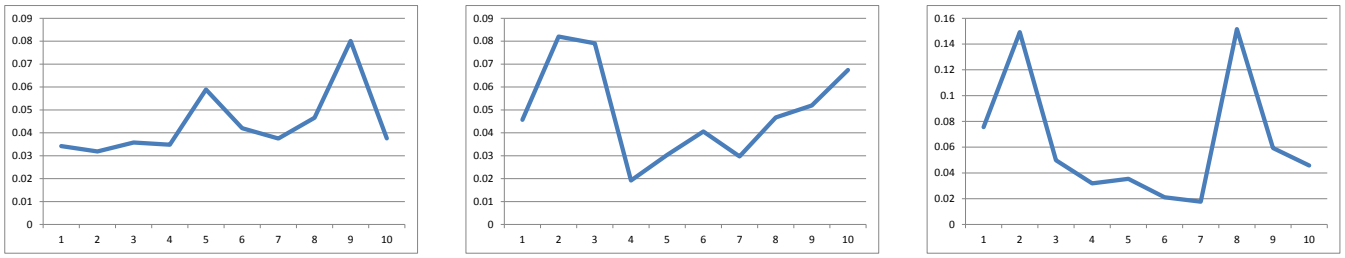
Regarding pre-processing steps, we performed four different steps on the source code. However, there is no consensus in the literature on which steps are essential or beneficial. Regarding parameter values, we used a well-established approach to find the optimal number of topics, which is much better than previous work in which they randomly selected a number of topics. We have focused on one open source Java-based systems. however, we cannot generalize the results. Additional case studies are needed to investigate closed-source and other programming languages systems.

VI. RELATED WORK

Mining software repositories is booming in the software engineering research community these recent years [16]–[18]. We discuss some of the related work on mining software repositories efficiently to help software maintenance tasks.

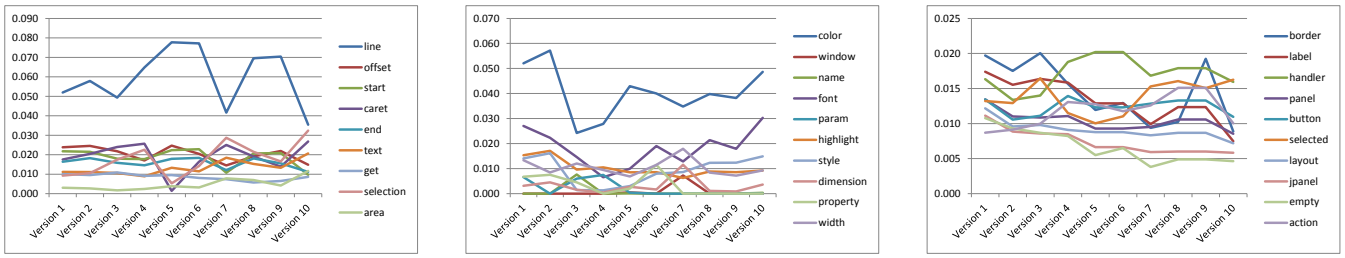
Sun et al. [19] proposed an approach based on LDA to find out what kind of historical information is needed to support software maintenance. They evaluated their approach by a new study on another important software maintenance task, i.e., feature location. Furthermore, their benchmarked their studies with more subject programs and metrics.

Herzig et al. [21] conducted empirical studies of tangled changes, which introduce noise in software repositories [20]. Their results were promising in which they showed that about



(a) Selection (b) Formatting (c) GUI

Figure 3. Topic Strength (Normalized Assignment).



(a) Selection (b) Formatting (c) GUI

Figure 4. Topic Content (TF values of top words within software versions).

20% of all bug fixes consist of multiple tangled changes. Keivanloo et al. proposed a collaborative platform for the purpose of sharing software datasets. Their platform supports data extraction, integration from various version control, issue tracking, and quality evaluation repositories. Their main focus was the integration of the information in software repositories.

Thomas et al. [4] proposed to use LDA to study the software evolution. They investigated whether the topics from the LDA corresponded well with actual code changes. Their results showed the effectiveness of using topic models as tools for studying the evolution of a software system. Furthermore, Their studies provided a good motivation for other researchers to use the topic model to mine the topics from software repositories.

VII. Conclusion

Available topic evolution models address only strength evolution or content evolution of the unstructured software repositories, not both like our approach. Having both the content evolution and the strength evolution will provide more comprehensive and complete results in order to understand the evolution of the source code than either one of them. In this work, we applied the Dynamic Topic Models to the source code to represent both their topic strength and content evolution. An empirical analysis of one well-known and open source projects, jEdit was conducted. We found that DTM produce complete and comprehensive view of software evolution, which is useful for a variety of stallholders to understand the changes of development topics from different aspects in a version. A future direction would be choosing more finest pre-processing steps. Another direction is to apply this approach on more software systems and conduct more comparative studies

Acknowledgment

The Authors would like to thank Prince Sultan University (PSU), Riyadh, K.S.A. for partially supporting this project.

References

[1] M. Alenezi, “Extracting high-level concepts from open-source systems,” International Journal of Software Engineering and Its Applications, vol. 9, no. 1, 2015, pp. 183–190.

[2] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, “How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms,” in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013, pp. 522–531.

[3] L. R. Biggers, C. Bocovich, R. Capshaw, B. P. Eddy, L. H. Etzkorn, and N. A. Kraft, “Configuring latent dirichlet allocation based feature location,” Empirical Software Engineering, vol. 19, no. 3, 2014, pp. 465–500.

[4] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, “Studying software evolution using topic models,” Science of Computer Programming, vol. 80, 2014, pp. 457–479.

[5] D. Hall, D. Jurafsky, and C. D. Manning, “Studying the history of ideas using topic models,” in Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, 2008, pp. 363–371.

[6] A. Hindle, M. W. Godfrey, and R. C. Holt, “What’s hot and what’s not: Windowed developer topic analysis,” in Software Maintenance, 2009. ICSM 2009. IEEE International Conference on. IEEE, 2009, pp. 339–348.

[7] Q. Mei and C. Zhai, “Discovering evolutionary theme patterns from text: an exploration of temporal text mining,” in Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, 2005, pp. 198–207.

[8] D. M. Blei and J. D. Lafferty, “Dynamic topic models,” in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 113–120.

 ISBN: 978-1-61208-438-1

[9] E. Linstead, C. Lopes, and P. Baldi, “An application of latent dirichlet allocation to analyzing software evolution,” in Seventh International Conference on Machine Learning and Applications, ICMLA 2008. IEEE, 2008, pp. 813–818.

[10] S. W. Thomas, B. Adams, A. E. Hassan, and D. Blostein, “Validating the use of topic models for software evolution,” in 10th IEEE Working Conference on Source Code Analysis and Manipulation (SCAM), 2010. IEEE, 2010, pp. 55–64.

[11] J. Hu, X. Sun, D. Lo, and B. Li, “Modeling the evolution of development topics using dynamic topic models,” in 22nd IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2015, pp. 3–12.

[12] R. Arun, V. Suresh, C. V. Madhavan, and M. N. Murthy, “On finding the natural number of topics with latent dirichlet allocation: Some observations,” in Advances in Knowledge Discovery and Data Mining. Springer, 2010, pp. 391–402.

[13] T. M. Cover and J. A. Thomas, Elements of information theory. John Wiley & Sons, 2012.

[14] T. L. Griffiths and M. Steyvers, “Finding scientific topics,” Proceedings of the National academy of Sciences of the United States of America, vol. 101, no. Suppl 1, 2004, pp. 5228–5235.

[15] S. Haiduc and A. Marcus, “On the use of domain terms in source code,” in The 16th IEEE International Conference on Program Comprehension, ICPC 2008. IEEE, 2008, pp. 113–122.

[16] K. Somasundaram and G. C. Murphy, “Automatic categorization of bug reports using latent dirichlet allocation,” in Proceedings of the 5th India Software Engineering Conference. ACM, 2012, pp. 125–130.

[17] M. Alenezi, K. Magel, and S. Banitaan, “Efficient bug triaging using text mining,” Journal of Software, vol. 8, no. 9, 2013, pp. 2185–2190.

[18] M. Alenezi and K. Magel, “Empirical evaluation of a new coupling metric: Combining structural and semantic coupling,” International Journal of Computers and Applications, vol. 36, no. 1, 2014.

[19] X. Sun, B. Li, Y. Li, and Y. Chen, “What information in software historical repositories do we need to support software maintenance tasks? an approach based on topic model,” in Computer and Information Science. Springer, 2015, pp. 27–37.

[20] K. Herzig and A. Zeller, “The impact of tangled code changes,” in 10th IEEE Working Conference on Mining Software Repositories (MSR). IEEE, 2013, pp. 121–130.

[21] I. Keivanloo, C. Forbes, A. Hmood, M. Erfani, C. Neal, G. Peristerakis, and J. Rilling, “A linked data platform for mining software repositories,” in 9th IEEE Working Conference on Mining Software Repositories (MSR). IEEE, 2012, pp. 32–35.

 ISBN: 978-1-61208-438-1

Model Transformation Applications from Requirements Engineering Perspective

Sobhan Yassipour Tehrani, Kevin Lano
Department of Informatics, King's College London, London WC2R 2LS, U.K.
E-mail: {sobhan.yassipour_tehrani,kevin.lano}@kcl.ac.uk

***Abstract*—Requirements Engineering (RE) is an essential process in the development of effective software systems, and it is the basis for subsequent development processes. At present, the focus of Model Transformation (MT) is mainly on the specification and implementation stages. Transformations are not using engineering principles, which may not be an issue within a small project, but it will be problematic in large scale industry projects. One of the main reasons that hinders a systematic RE process to be used before starting the development could be the false assumption that it is a waste of time/cost and would delay the implementation. The goal of this paper is to evaluate model transformation technology from a requirements engineering process point of view. We identify techniques for the RE of MT, taking into account specific characteristics of different categories of model transformations.**

Keywords- model transformations; requirements engineering; requirements engineering framework.

I. INTRODUCTION

Requirements engineering has been a relatively neglected aspect of model transformation development because the emphasis in transformation development has been upon specifications and implementations. The failure to explicitly identify requirements may result in developed transformations, which do not satisfy the needs of the users of the transformation. Problems may arise because implicitly-assumed requirements have not been explicitly stated; for instance, that a migration or refactoring transformation should preserve the semantics of its source model in the target model, or that a transformation is only required to operate on a restricted range of input models. Without thorough requirements elicitation, important requirements may be omitted from consideration, resulting in a developed transformation which fails to achieve its intended purpose.

We use the RE process model proposed by Kotonya and Sommerville [1] and adapt it according to our specific needs. This process model is widely accepted by researchers and professional experts. The following are the most important phases of RE, which have to be applied: domain analysis and requirements elicitation, evaluation and negotiation, specification and documentation, validation and verification.

In this paper we focus on the specification stage, which makes precise the informal requirements agreed with the stakeholders of the proposed development. By providing a comprehensive catalogue of model requirement types, this paper can help transformation developers to ensure that all requirements of a transformation are explicitly considered.

Section 3 gives a background on requirements engineering for model transformations as well as transformation semantics and its nature. We also identify how formalised requirements can be validated and can be used to guide the selection of design patterns for the development of the transformation. In Section 4 we examine some published requirements statements of model transformation to identify their gaps and subsequent consequences on the quality of the solutions. In Section 5 we give a case study to illustrate the benefits of systematic requirements engineering for model transformations.

II. STATE OF THE ART

As Selic [2] argues, "we are far from making the writing of model transformations an established and repeatable technical task". The software engineering of model transformations has only recently been considered in a systematic way, and most of this work [3][4][5] is focussed upon design and verification rather than upon requirements engineering. The work on requirements engineering in *trans*ML [3] is focussed upon functional requirements, and the use of abstract syntax rules to express them. Here, we consider a full range of functional and non-functional requirements and we use concrete syntax rules for the initial expression of functional requirements.

In order to trace the requirements into subsequent steps, *trans*ML defines a modelling language, which represents the requirements in the form of Systems Modeling Language (SysML) [6] diagrams. This would allows the transformer(s) to link requirements of a model transformation to its corresponding analysis and design models, code and other artifacts. Having a connection amongst different artifacts in the model transformation development process enables the transformer(s) to check the correctness and completeness of all requirements [7]. At present, transformations are not using engineering principles which may not be an issue within a small project, but it will be problematic in large scale industry projects. Jumping straight to an implementation language might be possible for simple transformations, however it would be problematic for large ones. Transformations should be constructed by applying engineering principles especially if they are to be used in an industry. Therefore, the development of the transformation's life-cycle should include other phases in addition to coding and testing, namely, requirements engineering process [3].

In this paper, we describe a requirements engineering process for transformations based on adaptions of the RE process model, and specialisations of RE techniques for transformations.

III. REQUIREMENTS FOR MODEL TRANSFORMATIONS

Requirements for a software product are generally divided into two main categories: functional requirements, which identify what functional capabilities the system should provide,

 ISBN: 978-1-61208-438-1

and non-functional requirements, which identify quality characteristics expected from the developed system and restrictions upon the development process itself.

The functional requirements of a model transformation τ:$S \rightarrow T$, which maps models of a source language S to a target language T are defined in terms of the effect of τ on model m of S, and the relationship of the resulting model n of T to m. It is a characteristic of model transformations that such functional requirements are usually decomposed into a set of mapping requirements for different cases of structures and elements within S. In addition, assumptions about the input model should be identified as part of the functional requirements.

It can be observed in many published examples of model transformations that the initial descriptions of their intended functional behaviour is in terms of a concrete syntax for the source and target languages, which they operate upon. For instance in [8], the three key effects of the transformation are expressed in terms of rewritings of Unified Modeling Language (UML) class diagrams. In [9], the transformation effects are expressed by parallel rewritings of Petri Nets and statecharts. In general, specification of the intended functionality of the transformation in terms of concrete syntax rules is more natural and comprehensible for the stakeholders than is specification in terms of abstract syntax. However, this form of description has the disadvantage that it may be imprecise; there may be significant details of models, which have no representation in the concrete syntax, or there may be ambiguities in the concrete syntax representation. Therefore, conversion of the concrete syntax rules into precise abstract syntax rules is a necessary step as part of the formalisation of the requirements.

Requirements may be functional or non-functional (e.g., concerned with the size of generated models, transformation efficiency or confluence). Another distinction, which is useful for transformations is between local and global requirements:

- Local requirements are concerned with localised parts of one or more models. Mapping requirements define when and how a part of one model should be mapped onto a part of another. Rewriting requirements dictate when and how a part of a model should be refactored/transformed in-place.
- Global requirements identify properties of an entire model. For example that some global measure of complexity or redundancy is decreased by a refactoring transformation. Invariants, assumptions and postconditions of a transformation usually apply at the entire model level.

Figure 1 shows a taxonomy of functional requirements for model transformations based on our experience of transformation requirements.

We have also created a taxonomy of the non-functional requirements that one has to consider during the RE process. Figure 2 shows a general decomposition of non-functional requirements for model transformations. The quality of service categories correspond closely to the software quality characteristics identified by the IEC 25010 software quality standard [10].

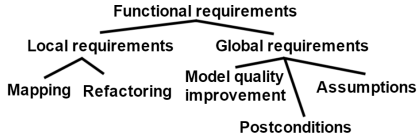


Figure. 1. A taxonomy of functional requirements

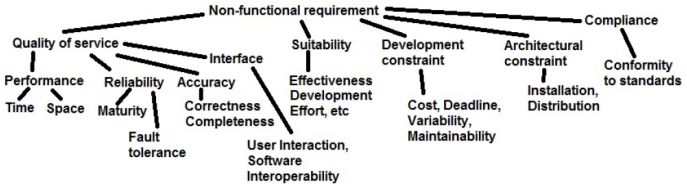


Figure. 2. A taxonomy of non-functional requirements for MT

Non-functional requirements for model transformations could be further detailed. For instance, regarding the performance requirements, boundaries (upper/lower) could be set on execution time, memory usage for models of a given size, and the maximum capability of the transformation (the largest model it can process within a given time). Restrictions can also be placed upon the rate of growth of execution time with input model size (for example, that this should be linear). Taxonomizing the requirements according to their type not only would make it clearer to understand what the requirements refer to, but also by having this type of distinction among them will allow for a more semantic characterization of requirements.

Maturity and fault tolerance are a subset of reliability requirements for a transformation. Depending on its history and to the extent to which a transformation has been used, maturity requirements could be measured. Fault tolerance requirements can be quantified in terms of the proportion of execution errors, which are successfully caught by an exception handling mechanism, and in terms of the ability of the transformation to detect and reject invalid input models.

As depicted in the above figure, the accuracy characteristic includes two sub-characteristics: correctness and completeness. Correctness requirements can be further divided into the following forms [11]:

- *Syntactic correctness*: a transformation τ is syntactically correct when a valid input model m from source language S is transformed to target language T, then (if it terminates) it produces a valid result, in terms of conformation to the T's language constraints.
- *Termination*: a transformation τ will always terminate if applied to a valid S model.
- *Confluence*: all result models produced by transformation τ from a single source model are isomorphic.
- *Model-level semantic preservation*: a transformation τ is preserved model-level semantically, if m and n have equivalent semantics under semantics-assigning maps Sem_S on models of S and Sem_T on models of T.

 ISBN: 978-1-61208-438-1

- *Invariance*: some properties *Inv* should be preserved as true during the entire execution of transformation τ [11].

An additional accuracy property that can be considered is the existence of invertibility in a transformation $\sigma : T \rightarrow S$ which inverts the effect of τ. Given a model n derived from m by τ, σ applied to n produces a model $m\prime$ of S isomorphic to m. A related property is change propagation which means that small changes to a source model can be propagated to the target model without re-executing the transformation. A further property of verifiability is important for transformations which is part of a business-critical or safety-critical process. This property identifies how effectively a transformation can be verified. Size, complexity, abstraction level and modularity are contributory factors to this property. The traceability property is the requirement that an explicit trace between mapped target model elements and their corresponding source model elements should be maintained by the transformation, and be available at its termination. Under interface are requirements categories of User interaction (subdivided into usability and convenience) and software interoperability. Usability requirements can be decomposed into aspects, such as understandability, learnability and attractiveness [12]. Software interoperability can be decomposed into interoperability capabilities of the system with each intended environment and software system, with which it is expected to operate.

Based on [12], we define suitability as the capability of a transformation approach to provide an appropriate means to express the functionality of a transformation problem at an appropriate level of abstraction, and to solve the transformation problem effectively and with acceptable use of resources (developer time, computational resources, etc.). In [8] we identified the following subcharacteristics for the suitability quality characteristic of model transformation specifications: abstraction level, size, complexity, effectiveness and development effort.

Requirements of single transformations can be documented using the SysML notation adopted in [3], but with a wider range of requirement types represented. Use case diagrams can be used to describe the requirements of a system of transformations. Each use case represents an individual transformation which may be available as a service for external users, or which may be used internally within the system as a subtransformation of other transformations.

We have investigated a specific functional requirements taxonomy according to the characteristic of model transformations (Table I). All types of functional requirements for model transformations including: mapping, assumptions and postconditions requirements could be formalized as predicates or diagrams at the concrete and abstract syntax levels. Concrete syntax is often used at the early stages (RE stages) in the development cycle in order to validate the requirements by stakeholders since the concrete syntax level is more convenient, whereas abstract syntax rule, is often used in the implementation phase for developers. However, there should be a direct correspondence between the concrete syntax elements in the informal/semi-formal expression of the requirements, and the abstract syntax elements in the formalised versions.

TABLE I. TRANSFORMATION REQUIREMENTS CATALOGUE

	Refactoring	**Refinement**	**Migration**
Local Functional	Rewrites/ Refactorings	Mappings	Mappings
Local Non-functional	Completeness(all cases considered)	Completeness (all source entities, features considered)	Completeness (all source entities, features considered)
Global Functional	Improvement in quality measure(s), Invariance of language constraints, Assumptions, Postconditions	Invariance, Assumptions, Postconditions	Invariance, Assumptions, Postconditions
Global Non-functional	Termination, Efficiency, Modularity, Model-level semantic preservation, Confluence, Fault tolerance, Security	Termination, Efficiency, Modularity, Traceability, Confluence, Fault tolerance, Security	Termination, Efficiency, Modularity, Traceability, Confluence, Fault tolerance

IV. APPLICATION OF RE IN MT

In model transformation, requirements and specifications are very similar and sometimes are considered as the same element. Requirements determine what is needed and what needs to be achieved while taking into account the different stakeholders, whereas specifications define precisely what is to be developed.

Requirements engineering for model transformations involves specialised techniques and approaches because transformations (i) have highly complex behaviour, involving nondeterministic application of rules and inspection/ construction of complex model data, (ii) are often high-integrity and business-critical systems with strong requirements for reliability and correctness.

Transformations do not usually involve much user interaction, but may have security requirements if they process secure data. Correctness requirements which are specific to transformations, due to their characteristic execution as a series of rewrite rule applications, with the order of these applications not algorithmically determined, are: (i) confluence (that the output models produced by the transformation are equivalent, regardless of the rule application orders), (ii) termination (regardless of the execution order), (iii) to achieve specified properties of the target model, regardless of the execution order which is referred to as semantic correctness.

The source and target languages of a transformation may be precisely specified by metamodels, whereas the requirements for its processing may initially be quite unclear. For a migration transformation, analysis will be needed to identify how elements of the source language should be mapped to elements of the target. There may not be a clear relationship between parts of these languages, there may be ambiguities and choices in mapping, and there may be necessary assumptions on the

 ISBN: 978-1-61208-438-1

input models for a given mapping strategy to be well-defined. The requirements engineer should identify how each entity type and feature of the source language should be migrated.

For refactorings, the additional complications arising from update-in-place processing need to be considered and the application of one rule to a model may enable further rule applications which were not originally enabled. The requirements engineer should identify all the distinct situations which need to be processed by the transformation such as arrangements of model elements and their inter-relationships and significant feature values.

A. Application of RE Techniques for MT

A large number of requirements elicitation techniques have been devised. Through the analysis of surveys and case studies, we have identified the following adaption of RE techniques for MT.

The following techniques are the most suitable RE techniques to use during the requirements elicitation stage, which have been adapted according to the nature of model transformation technology.

Structured interviews: in this technique the requirements engineer asks stakeholders specific prepared questions about the domain and the system. The requirements engineer needs to define appropriate questions which help to identify issues of scope and product (output model) requirements, similar to that of unstructured interviews. This technique is relevant to all forms of transformation problems. We have defined a catalogue of MT requirements for refactorings, refinements and migrations, as an aid for structured interviews, and as a checklist to ensure that all forms of requirements appropriate for the transformation are considered.

Rapid prototyping: in this technique a stakeholder is asked to comment on a prototype solution. This technique is relevant for all forms of transformation, where the transformation can be effectively prototyped. Rules could be expressed in a concrete grammar form and reviewed by stakeholders, along with visualisations of input and output models. This approach fits well with an Agile development process for transformations.

Scenario analysis: in this approach the requirements engineer formulates detailed scenarios/use cases of the system for discussion with the stakeholders. This is highly relevant for MT requirements elicitation. Scenarios can be defined for different required cases of transformation processing. The scenarios can be used as the basis of requirements formalisation. This technique is proposed for transformations in [3]. A risk with scenario analysis is that this may fail to be complete and may not cover all cases of expected transformation processing. It is more suited to the identification of local rather than global requirements.

Regarding the requirements evaluation and negotiation stage, prototyping techniques are useful for evaluating requirements, and for identifying deficiencies and areas where the intended behaviour is not yet understood. A goal-oriented analysis technique such as Knowledge Acquisition in automated specification (KAOS) or SySML can be used to decompose requirements into sub-goals. A formal modelling notation such as Object Constraint Language (OCL) or state machines/state charts can be used to expose the implications of requirements. For transformations, state machines may be useful to identify implicit orderings or conflicts of rules which arise because the effect of one rule may enable or disable the occurrence of another. Requirements have to be prioritized according to their importance and the type of transformation. For instance, in a refinement transformation, the semantics of the source and target model have to be equivalent as the primary requirement and to have a traceability feature as a secondary requirement. Also, there should be no conflict among the requirements. For instance, there is often a conflict between the time, quality and budget of a project. The quality of the target model should be satisfactory with respect to the performance (time, cost and space) of the transformation. Several RE techniques exist which could be applicable to the transformation of RE during the requirements specification phase in which business goals are represented in terms of functional and non-functional requirements. In the following Table 2, requirements have been categorised according to the type of the transformation.

TABLE II. REQUIREMENTS PRIORITY FOR DIFFERENT TRANSFORMATIONS

Category	**Primary requirement**	**Secondary requirement**
Refactoring	Model quality improvement Model-level semantic preservation Syntactic correctness Termination	Invariance Confluence
Migration	Syntactic correctness Model-level semantic preservation Termination	Invertibility Confluence Traceability
Refinement	Syntactic correctness Model-level semantic preservation Confluence Termination	Traceability

Techniques for requirements specification and documentation stage include: UML and OCL, structured natural language, and formal modelling languages. At the initial stages of requirements elicitation and analysis, the intended effect of a transformation is often expressed by sketches or diagrams using the concrete grammar of the source and target languages concerned (if such grammars exist), or by node and line graphs if there is no concrete grammar. A benefit of concrete grammar rules is that they are directly understandable by stakeholders with knowledge of the source and target language notations. They are also independent of specific MT languages or technologies. Concrete grammar diagrams can be made more precise during requirements formalisation, or refined into abstract grammar rules. An informal mapping/refactoring requirement of the form of

> "*For each instance e of entity type E, that satisfies condition Cond, establish Pred* "

can be formalised as a use case postcondition such as:

 ISBN: 978-1-61208-438-1

E::
Cond′ ⇒ Pred′

where *Cond′* formalises Cond, and *Pred′* formalises Pred.

For requirements verification and validation stage, the formalised rules can be checked for internal correctness properties such as definedness and determinacy, which should hold for meaningful rules. A prototype implementation can be generated, and its behaviour on a range of input models covering all of the scenarios considered during requirements elicitation can be checked. When a precise expression of the functional and non-functional requirements has been defined, it can be validated with the stakeholders to confirm that it does indeed accurately express the stakeholders intentions and needs for the system. The formalised requirements of a transformation τ: $S \rightarrow T$ can also be verified to check that they are consistent; the functional requirements must be mutually consistent. The assumptions and invariant of τ, and the language constraints of S must be jointly consistent. The invariant and postconditions of τ, and the language constraints of T must be jointly consistent. Each mapping rule Left-Hand Side (LHS) must be consistent with the invariant, as must each mapping rule Right-Hand Side (RHS).

These consistency properties can be checked using tools such as Z3 or Alloy, given suitable encodings [13], [14]. Model-level semantics preservation requirements can in some cases be characterised by additional invariant properties which the transformation should maintain. For each functional and non-functional requirement, justification should be given as to why the formalised specification satisfies these requirements. For example, to justify termination, some variant quantity Q : Integer could be identified which is always non-negative and which is strictly decreased by each application of a mapping rule [11]. Formalised requirements in temporal logic could then be checked for particular implementations using model-checking techniques, as in [15].

V. RE PROCESS ON REFACTORING TRANSFORMATION

Refactoring is a type of model transformation. The general idea behind refactoring is to improve the structure of the model to make it easier to understand, and to make it more maintainable and amenable to change. According to Fowler, refactoring could be defined as "changing a software system in such a way that it does not alter the external behaviour of the code, yet improves its internal structure" [16]. We describe an example [17] of an in-place endogenous transformation which refactors class diagrams to improve their quality by removing redundant feature declarations. Figure 3 shows the metamodel of the source/target language of this transformation.

In this section, we are going to apply RE on a refactoring [18] transformation case study. The properties for this type of transformation are: endogenous, model-to-model, many-to-many (source to target model), horizontal, semantics preservation, explicit control/rule application scoping, rule iteration, traceable and that it is a unidirectional transformation. The following general requirements for refactoring transformations should be satisfied:

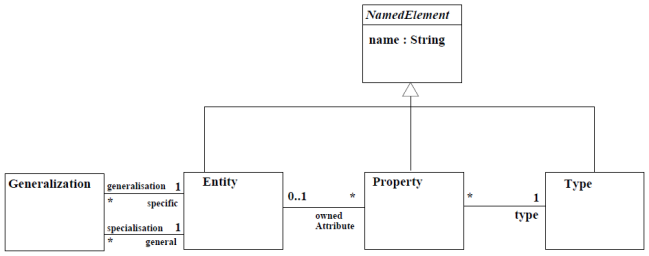


Figure. 3. Class diagram metamodel [18]

- Functionality: suitability, accuracy, interoperability, security, functionality compliance
- Reliability: maturity, fault tolerance, recoverability, reliability compliance
- Usability: understandability, learnability, operability, attractiveness, usability compliance
- Efficiency: time behaviour, resource utilisation, efficiency compliance
- Maintainability: analysability, changeability, stability, testability, maintainability compliance
- Portability: adaptability, installability, co-existence, replaceability, portability compliance

1) Requirements elicitation for Refactoring: The initial requirements statement is to refactor a UML class diagram to remove all cases of duplicated attribute declarations in sibling classes (classes which have a common parent). This statement is concerned purely with functional behaviour. Through structured interviews with the customer (and with the end users of the refactored diagrams and the development team) we can further uncover, non-functional requirements as follows: *efficiency*, the refactoring should be able to process diagrams with 1000 classes and 10,000 attributes in a practical time (less than 5 minutes), *correctness*, the start and end models should have equivalent semantics, *minimality*: the number of new classes introduced should be minimized to avoid introducing superfluous classes into the model, *confluence*, would be desirable but is not mandatory.

The functional requirements can also be clarified and more precisely scoped by the interview process. A global functional requirement is the invariance of the class diagram language constraints meaning that there is no multiple inheritance, and no concrete class with a subclass. It is not proposed to refactor associations because of the additional complications this would cause for the developers. Only attributes are to be considered. Through scenario analysis using concrete grammar sketches, the main functional requirement is decomposed into three cases: (i) where all (two or more) direct subclasses of one class have identical attribute declarations, (ii) where two or more direct subclasses have identical attribute declarations, (iii) where two or more root classes have identical attribute declarations.

2) Evaluation and negotiation for Refactoring: At this point we should ask whether these scenarios are complete and if they cover all intended cases of the required refactor-

 ISBN: 978-1-61208-438-1

ings. Through the analysis of the possible structures of class diagrams, and by taking into account the invariant of single inheritance, it can be deduced that they are complete. Through exploratory prototyping and execution on particular examples of class diagrams, we can identify that the requirement for minimality means that rule 1 Pull up attributes should be prioritised over rule 2 Create subclass or 3 Create root class. In addition, the largest set of duplicated attributes in sibling classes should be removed.

3) Requirements formalisation for Refactoring: To formalise the functional requirements, we express the three scenarios in the abstract grammar of the language. Rule1: If the set g = *c.specialisation.specific* of all direct subclasses of a class c has two or more elements, and all classes in g have an owned attribute with the same name n and type t, add an attribute of this name and type to c, and remove the copies from each element of g. Rule 2: If a class c has two or more direct subclasses g = *c.specialisation.specific*, and there is a subset *g1* of g, of size at least 2, all the elements of *g1* have an owned attribute with the same name n and type t, but there are elements of g - *g1* without such an attribute, introduce a new class *c1* as a subclass of c. *c1* should also be set as a direct superclass of all those classes in g which own a copy of the cloned attribute. Add an attribute of name n and type t to *c1* and remove the copies from each of its direct subclasses. Rule 3: If there are two or more root classes all of which have an owned attribute with the same name n and type t, create a new root class c. Make c the direct superclass of all root classes with such an attribute, and add an attribute of name n and type t to c, and remove the copies from each of the direct subclasses.

4) Validation and verification for Refactoring: The functional requirements can be checked by executing the prototype transformation on test cases. In addition, informal reasoning can be used to check that each rule application preserves the invariants. For example, no rule introduces new types, or modifies existing types, so the invariant that type names are unique is clearly preserved by rule applications. Likewise, the model-level semantics is also preserved. Termination follows by establishing that each rule application decreases the number of attributes in the diagram, i.e., *Property.size*. The efficiency requirements can be verified by executing the prototype transformation on realistic test cases of increasing size.

VI. Conclusion and Future Work

We have identified ways in which requirements engineering can be applied systematically to model transformations. Comprehensive catalogues of functional and non-functional requirements categories for model transformations have been defined. We have examined a case study which is typical of the current state of the art in transformation development, and identified how formal treatment of functional and non-functional requirements can benefit such developments. In future work, we will construct tool support for recording and tracing transformation requirements, which will help to ensure that developers systematically consider all necessary requirements and that these are all formalised, validated and verified correctly.

We are currently carrying out research into improving the requirements engineering process in model transformation. We will investigate formal languages to express the requirements, as formalised rules can be checked for internal correctness properties, such as definedness and determinacy, which should hold for meaningful rules. Temporal logic can be used to define the specialised characteristics of particular transformation and to define transformation requirements in a formal but language-independent manner languages as model transformation systems necessarily involve a notion of time. Finally, we will be evaluating large case studies in order to compare results with and without RE process.

References

[1] I. Sommerville and G. Kotonya, Requirements engineering: processes and techniques. John Wiley & Sons, Inc., 1998.

[2] B. Selic, "What will it take? a view on adoption of model-based methods in practice," Software & Systems Modeling, vol. 11, no. 4, 2012, pp. 513–526.

[3] E. Guerra, J. De Lara, D. S. Kolovos, R. F. Paige, and O. M. dos Santos, "transml: A family of languages to model model transformations," in Model Driven Engineering Languages and Systems. Springer, 2010, pp. 106–120.

[4] K. Lano and S. Kolahdouz-Rahimi, "Model-driven development of model transformations," in Theory and practice of model transformations. Springer, 2011, pp. 47–61.

[5] K. Lano and S. Rahimi, "Constraint-based specification of model transformations," Journal of Systems and Software, vol. 86, no. 2, 2013, pp. 412–436.

[6] S. Friedenthal, A. Moore, and R. Steiner, A practical guide to SysML: the systems modeling language. Morgan Kaufmann, 2014.

[7] T. Yue, L. C. Briand, and Y. Labiche, "A systematic review of transformation approaches between user requirements and analysis models," Requirements Engineering, vol. 16, no. 2, 2011, pp. 75–99.

[8] S. Kolahdouz-Rahimi, K. Lano, S. Pillay, J. Troya, and P. Van Gorp, "Evaluation of model transformation approaches for model refactoring," Science of Computer Programming, vol. 85, 2014, pp. 5–40.

[9] P. Van Gorp and L. M. Rose, "The petri-nets to statecharts transformation case," arXiv preprint arXiv:1312.0342, 2013.

[10] I. Iso, "Iec 25010: 2011,," Systems and Software EngineeringSystems and Software Quality Requirements and Evaluation (SQuaRE)System and Software Quality Models, 2011.

[11] K. Lano, S. Kolahdouz-Rahimi, and T. Clark, "Comparing verification techniques for model transformations," in Proceedings of the Workshop on Model-Driven Engineering, Verification and Validation. ACM, 2012, pp. 23–28.

[12] I. O. F. S. E. Commission et al., "Software engineering–product quality–part 1: Quality model," ISO/IEC, vol. 9126, 2001, p. 2001.

[13] K. Anastasakis, B. Bordbar, and J. M. Küster, "Analysis of model transformations via alloy," in Proceedings of the 4th MoDeVVa workshop Model-Driven Engineering, Verification and Validation, 2007, pp. 47–56.

[14] L. de Moura and N. Bjørner, "Z3–a tutorial," 2006.

[15] S. Yassipour Tehrani and K. Lano, "Temporal logic specification and analysis for model transformations," in Verification of Model Transformations, VOLT 2015, 2015.

[16] C. Ermel, H. Ehrig, and K. Ehrig, "Refactoring of model transformations," Electronic Communications of the EASST, vol. 18, 2009.

[17] K. Lano and S. K. Rahimi, "Case study: Class diagram restructuring," in Proceedings Sixth Transformation Tool Contest, TTC 2013, Budapest, Hungary, 19-20 June, 2013., 2013, pp. 8–15.

[18] S. Kolahdouz-Rahimi, K. Lano, S. Pillay, J. Troya, and P. Van Gorp, "Evaluation of model transformation approaches for model refactoring," Science of Computer Programming, vol. 85, 2014, pp. 5–40.

 ISBN: 978-1-61208-438-1

Analyzing the Evolvability of Modular Structures: a Longitudinal Normalized Systems Case Study

Philip Huysmans, Peter De Bruyn, Gilles Oorts,
Jan Verelst, Dirk van der Linden and Herwig Mannaert

Normalized Systems Institute
University of Antwerp
Antwerp, Belgium
Email: {philip.huysmans,peter.debruyn,gilles.oorts,
jan.verelst,dirk.vanderlinden,herwig.mannaert}@uantwerp.be

***Abstract*—The evolvability of organizations as a whole is determined by the evolvability of different enterprise architecture layers. This paper presents a longitudinal case study, performed within an infrastructure monitoring company, on how Normalized Systems Theory enables this evolvability, at least, at the level of its information systems. By describing the different versions of the case organization's information system throughout time, we are able to analyze the characteristics of the system which facilitate this goal. In particular, the increasingly fine-grained structure of the system allows for multiple dimensions of variability. This analysis is then generalized and described in terms of modular systems. Based on this generalization, several implications for other enterprise layers are presented.**

Keywords–Normalized Systems; modularity; evolvability; case study

I. INTRODUCTION

In today's ever-changing and competitive markets, enterprises need to be able to respond ever so quickly to changing market demands. One could argue that this evolvability is one of the main requirements for an enterprise to be competitive in the current global economy. Certain scholars have argued that in order to create a sustainable competitive advantage, changes need to be applied *at a constant rate* [1]. This, however, means an organization is in a constant state of flux, and a fixed baseline on which new changes can be etched is ever absent. This considerably complicates the implementation of changes and the agility of organizations.

An additional challenge to organizational evolvability is that evolvability is required at multiple enterprise layers. For example, to attain a truly evolvable enterprise, its organizational structure, business processes and information systems need to be able to easily implement changes. As all these layers are intertwined, a single change in one of these layers will insurmountably result in multiple changes in one or more of the other layers. As a result, it is clear one should always study organizational evolvability as the accumulation of evolvability within all these inseparable layers. Likewise, enterprises should always strive for organizational evolvability within all layers.

Most enterprise architecture approaches propose a generic way of working towards evolvability. For example, Ross et al. [2] propose that, after a team and vision are established, an AS-IS architecture is developed. Next, a TO-BE architecture should be defined which enables the established vision. The transition between AS-IS and TO-BE architectures then needs to be planned and executed. Most frameworks do not provide a more concrete way of working [3].

At the lower organizational levels however, more detailed progress has been made in enabling evolvability. This is especially true for the lowest level, i.e., the information systems that support the other enterprise layers in the execution of their tasks. In this regard, Normalized Systems (NS) theory was introduced as an approach to build evolvable artifacts, such as information systems [4]. Although this approach is proven to be theoretically sound [5] and practically viable [6], [7], few cases have been published.

In this paper, we will therefore document a case to illustrate how Normalized information systems support organizational evolvability by allowing rapid and extensive changes to the software. The specific case is chosen because it concerns a software application that evolved extensively throughout time and allows to clearly illustrate why NS theory requires a fine-grained modular structure. NS theory has also proven to be relevant to design artifacts in other organizational layers as well [3], [8], [9]. Therefore, we will generalize the findings and reflect on the potential implications for modular structures in later sections of the paper.

The paper is structured as follows: first we introduce the Normalized Systems theory in Section II. Next, we describe the case study and the evolution of the discussed application in detail in Section III. The case reflections, generalizations and implications are discussed in the Discussion in Section IV, followed by a Conclusion in Section V.

II. NORMALIZED SYSTEMS

The case that is discussed in this paper is based on the body of thought of Normalized Systems (NS) theory. Therefore, we will briefly introduce this theory in this section. For a more comprehensive description, we refer to previous publications, such as [4], [5], [10], [11].

The NS theory is theoretically founded on the concept of stability from systems theory. According to systems theory, stability is an essential property of systems. For a system to be stable, a bounded input should result in a bounded output, even if an unlimited time period $T \rightarrow \infty$ is considered. For

 ISBN: 978-1-61208-438-1

information systems, this means that a bounded set of changes (selected from the so-called *anticipated changes* within NS theory) should result in a bounded impact to the system, even for $T \rightarrow \infty$ (i.e., an unlimited systems evolution is considered). In other words, stability reasoning expresses how the impact of changes to an information system should not depend on the size of the system, but only on size and property of the changes that need to be performed. If this is not the case, a so-called *combinatorial effect* occurs. It has been formally proven that any violation of any of the following theorems will result in combinatorial effects that negatively impact evolvability [5]:

- *Separation of Concerns*, which states that each concern (i.e., each change driver) needs to be encapsulated in an element, separated from other concerns;
- *Action Version Transparency*, which declares an action entity should be updateable without impacting the action entities it is called by;
- *Data Version Transparency*, which indicates a data entity should be updateable without impacting the action entities it is called by;
- *Separation of States*, which states all actions in a workflow should be separated by state (and called in a stateful way).

The application of the NS theorems in practice has shown to result in very fine-grained modular elements which may, at first, be regarded as complex. Although it quickly becomes clear to developers how every element is constructed very similarly, it is very unlikely to attain these strictly defined elements without the use of higher-level primitives or patterns. Therefore NS theory proposes a set of five elements (action, data, workflow, connector and trigger) that serve as patterns. Based on these elements, NS software is generated in a relatively straightforward way through the use of the NS expansion mechanism. For this purpose, dedicated software (called NS expanders) was built by the Normalized Systems eXpanders factory (NSX).

III. Case Study

The case we discuss in this paper is that of an organization which provides hardware and software for infrastructure monitoring (e.g., power supplies, air conditioning, fire detection systems and diesel generators). The infrastructure is monitored and managed from a central site called the Network Operating Center (NOC). In this center, status information from different facility equipment of geographically dispersed sites is gathered. The status information is sent by controllers which are embedded in the infrastructure. Different types of these controllers are developed and marketed by the organization. For example, the Telecom Site Controller (TSC) is developed specifically for telecom infrastructure, and the Monitoring Control Unit (MCU) is developed specifically for DC power supplies which contain AC/DC converters and batteries to handle power failures. The organization argued that the software to perform and manage the infrastructure monitoring could not be purchased as a commercial off-the-shelf package, because of the extensive customizations needed for the proprietary controllers and protocols. Consequently, a custom application was developed. We will describe the evolution of this application as a longitudinal case study by means of four phases the system has gone through.

A. Phase 1: SMS v1

Functionality and technology: The original version of the Site Management System (which we refer to as SMS v1) was deployed at the organization itself for a client from the railroad sector, as well as on-site for different clients from, a.o., the fiber glass sector. Initially, SMS v1 only supported the proprietary TSC controllers developed by the organization itself. Later on, MCU controllers were added, but only for certain clients. After initial deployment, more sites were added, and the application provided monitoring of around 280 sites.

SMS v1 was developed using Visual Basic and MS Access technology. The lack of a client-server architecture in the technology stack forced the organization to adopt suboptimal solutions for using the system in a distributed way: the MS Access database was remotely accessed through a network drive, and remote usage of the Visual Basic application was done by using a Virtual Network Computing (VNC) connection.

Structure: An explicit and deliberate structure of the application did not exist. Rather, the application was regarded as one single monolithic module, which relied on a database module. As a result, no code reuse was present: application code was duplicated in an unstructured way for every deployment instance.

Evolvability: The coarse-grained structure of the application resulted in certain quality deficiencies. The evolvability of SMS v1 was hard to determine: changes made to a certain code base were not always introduced in other code bases, which resulted in distinct (inconsistent) code bases. As a result, any change could have a different impact on a specific SMS v1 code base.

The arduousness of changing the code base was aggravated by the amount of configuration parameters which were hard-coded. For example, it was assumed that all TSCs in a network were configured in the same way (e.g., the alarms from the air conditioning system are registered on input 1). This resulted in a lack of flexibility during deployment. Moreover, the growth, increased usage, and multi-user access of the application resulted in performance issues. The used technology was not designed for scalability, and was focused on single-user access.

B. Phase 2: SMS v2

Functionality and technology: As the initial version of SMS provided adequate functionality, the need for a new version was largely motivated by non-functional requirements. As reported above, the scalability and flexibility of the original application was unsatisfactory. In 2005, an external software development company was approached. Based on the existing functionality, a new application was developed from scratch and deployed in 2006. We will refer to this application as Site Management System version 2 (SMS v2).

Because the lack of a client-server architecture was experienced as an obstacle for a scalable, flexible and multi-user application, a radical change of the application architecture was adopted. Instead of using the proprietary Microsoft technologies of SMS v1, a standard and web-based architecture was adopted. More specifically, the Java 2 Enterprise Edition (J2EE) platform was used, with Enterprise JavaBeans version 2.1 (EJB2) and Cocoon framework as characterizing components.

 ISBN: 978-1-61208-438-1

Structure: SMS v2 was developed following industry best practices, which imposed a certain structure: different concerns need to be implemented in different constructs (e.g., java classes). For example, EJB2 prescribes that for each bean, local and home interfaces need to be defined, and that RMI-access to the bean must be provided by an agent class. Similarly, a certain structure was imposed by the Cocoon framework, which was employed for the web tier of the application. The actual business logic (e.g., checking the status of a TSC controller) was also implemented by separate classes. This prevents the inclusion of business logic in framework-specific classes: the controller class contains the actual description of the controller, such as the IP address and port, protocol, or phone number for dial-up access. Consequently, the parameters for each controller were clearly separated from other concerns, and could be configured separately, providing the required flexibility.

This way of working leveraged existing knowledge in the software engineering field, which is distributed in several ways. First, design patterns describe generally accepted solutions on how code should be structured in certain situations. For example, the Strategy pattern from the Gang of Four pattern catalog describes a structure to implement a "family of algorithms", and make them interchangeable [12, p. 315]. This structure was applied in the SMS v2 application, and enabled the loading of the correct implementation class of a specific controller (e.g., TSC or MCU). Second, the usage of frameworks enforces certain industry best practices. For example, the Model-View-Controller design pattern [13] can be implemented in any object-oriented language by the programmer. Frameworks such as Cocoon enforce programmers to adhere to this pattern, thus eliminating a certain design freedom. Similarly, the usage of EJB2 also encourages a programmer to separate certain concerns. For example, by using object-relational mapping, a separation between logic and persistence is enforced.

These examples illustrate how applying existing software engineering knowledge enabled non-functional requirements such as flexibility and scalability by prescribing a finer-grained structure of software constructs. However, the application exhibited an even finer-grained structure than prescribed by the design patterns and frameworks. For example, a specific class was created to trigger certain tasks at certain intervals (e.g., checking if an alarm was generated). Separating this functionality is not prescribed by design patterns or frameworks: by considering it as business logic, it could be included in the implementation classes. Nevertheless, separating this rather generic functionality in its own constructs allowed the reuse its code in different contexts. As a result, the structure of software primitives for various entities started to exhibit a similar structure.

Evolvability: While developing SMS v2, NS theory was not yet formulated and the expanders were not yet developed. As a result, the recurring code structure (which contained many constructs) needed to be recreated manually. While still requiring some effort, this was relatively easy as each particular controller was rather similar.

Using a recurring structure resulted in other advantages as well. Due to experience with similar code structures used for other controllers (or even, similar code structures in other applications), the performance of the application under different loads (e.g., number of status messages sent) could be accurately estimated. As a result, scaling the application across different sites could be managed.

C. Phase 3: PEMM v1

Functionality and technology: Around 2007, SMS needed to support new functional requirements. First, the range of supported controllers was to be extended. For example, support was added for OLE for Process Control (OPC) servers. An OPC server groups communication from multiple controllers, which allows easier hardware setup. Second, specific functionality for certain controller types was to be supported. For example, a TSC controller provides configuration management for physical site access control. By sending configuration messages, access codes for specific sites with keypad access can be set. Third, various output options were to be provided. In case of certain alerts, an SMS could be sent to the operator, in addition to the regular monitor-based output. Because of the size of the new functional requirements, the application was renamed in Power Environmental Monitoring and Management (PEMM). We will refer to it as PEMM v1.

The technology stack of PEMM v1 was similar to the technology stack of SMS v2: a J2EE architecture with EJB2 and Cocoon framework. The versions of the different components were updated to more up-to-date versions.

Structure: The consistent and systematic separation of different concerns in several projects had resulted in a recurring software structure. For example, gathering and persisting data for certain entities required several constructs for creating a Create-Read-Update-Delete-Search (CRUDS) interface (i.e., jsp and html pages), java classes for the application server, and relation database table specifications. As a result, the required constructs in use could be reused for every new instance. In order to facilitate this reuse, a set of *pattern expanders* was created, which create the software constructs based on a configuration file. For example, the constructs for the data entities are created by the data element pattern expander. Parameters for the data entity are specified in a XML configuration file, also called a *descriptor file*. For a data element, for following parameters need to be defined:

- Basic name of the data element instance.
- Context information (i.e., package and component name)
- Data field information (i.e., names and data types for the various attributes of the entity)
- Relationships with other elements

For the PEMM v1 application, such descriptor files were created for, e.g., controllers, alarms, sites, etc.

After such pattern expansion, the application can be compiled and deployed, similar to a regular application. Pattern expansion allows developers to quickly create a software structure which separates many concerns. Developing such structure from scratch would imply a disproportionate effort when compared to the effort required for programming the actual business logic. As a result, separating many concerns is often omitted, which results in code of poorer quality. Such pattern expansion is only feasible when every data element has an identical structure.

Evolvability: Because of identical structures within the code base, applying changes to the code becomes predictable, or even deterministic. We discuss four main groups of changes.

 ISBN: 978-1-61208-438-1

First, functional changes could be added to existing elements by *marginal expansion*. For example, adding a data attribute for a controller could be specified in an additional data descriptor. A marginal expansion recognizes the element for which the additional descriptor is specified, and adds the necessary code in the existing code base. As a result, certain functional changes can be made without overwriting customizations, and are coined *anticipated changes* [10, p. 95]:

- an additional data field;
- an additional data entity;
- an additional action entity;
- an additional version of a task.

In PEMM v1, an example of a marginal expansion was the addition of a comment field to an alarm data element. Adding the comment field through marginal expansion not only adds a field in the database table, but also adds that field in the java bean, in all CRUDS screens, etc.

Second, new functionality can be added to the application by *generating new elements*. These can be integrated in the existing code base by providing relations to the existing elements in the descriptor files. In PEMM v1, the following functionality was added by expanding new elements into the existing application:

- FAQs: a FAQ element allows customers to input knowledge concerning specific alarms. These FAQs are made available to operators who need to monitor and manage the alarms.
- Asset management: in order to keep track of various assets, an asset element was added to allow a technician to add the serial number of used or newly added assets to a certain site.
- Service log: a service log element records the history of all service interventions made on a particular site.

Third, functionality which cannot be implemented by expanding new elements or by applying anticipated changes, needs to be realized through *customizations*. Non-standard functionality, such as user interface screens, reporting or authentications needs to be programmed separately. Implementation classes for actions (e.g., checking a controller) are typical examples of such customizations, which are added to the code base as separate files. Another example for PEMM v1 is the reporting functionality, which is an implementation class for an action element which generates a file to import alarm data in reporting tools. Separate files which are necessary in the code base (such as the implementation classes) can be easily located by programmers, since they always occur at the same location within the element structure. However, customizations can also be made by overwriting code in the generated files. Such customizations are harder to track, as white-box inspection or separate documentation is required to know where they are located. In PEMM v1, the following customizations in the expanded code needed to be made:

- Authorization: in an NS application, a base component is added to configure user authorizations. In the PEMM application, custom business rules were added based on these configurations. For example, certain users could acknowledge collections of alarms, instead of acknowledging each alarm separately. In multi-tenant deployments of PEMM, this ability needed to be restricted to alarms from certain sites.
- User Interface: examples of user interface screens which were added to the PEMM application as customization are: (1) trending charts, which show certain measurements over time; (2) Alarm overview screens: color-coded tables which provide a high-level view of active alarms; (3) Map view: a map which shows sites with active alarms. This map view is linked to the alarm overview screens.

D. Phase 4: PEMM v2

Functionality and technology: Around 2012-2013, the organization decided to switch from the proprietary TSC and MCU controllers and protocols towards industry-standard controllers (Beckoff) and protocols (SNMP, Modbus over IP and OPC DCOM DA). The introduction of these requirements triggered an update of the PEMM application using newer versions of the NS expanders (now called PEMM v2), which incorporated a new mechanism to facilitate the extraction (*harvesting*) and addition (*injection*) of customizations. While the functional changes could have been implemented in PEMM v1 without combinatorial effects, they required customizations which would need to be redone in the case of a regeneration at a later point in time. Therefore, the decision was made to migrate to the new version of expanders. As the new version of expanders can use the same descriptor files and expand a similar source code structure, porting the application to a new expander version required much less effort in comparison with a rewrite. Using the new expander versions also implied the incorporation of new frameworks (e.g., Knockout) and new versions of the existing frameworks (e.g., Struts 2), compilers (JDK) and servers (e.g., Jonas application server) with their accompanying fixes and functional enhancements. Moreover, the programming team changed: a developer of the infrastructure monitoring organization, familiar with these new frameworks, was appointed to work on the new application.

Structure:

Again, a large portion of the code base could be generated using the expanders resulting in a very similar general structure as before. This time however, *anchors* were added to the structure, which allow the harvesting/injection mechanism to work. This mechanism solves the issue of overwriting customizations in case of regeneration. Customizations to the code base (e.g., GUI elements) can now be made in three ways:

- insertions: customizations are put between predefined anchors in the expanded code base;
- extensions: customizations are contained within separate files, added in the file structure in predefined directories;
- overlays (discouraged): customizations overwriting expanded files, but not being captured by the harvesting/injection mechanism.

In case of a regeneration, the harvesting mechanism checks the anchors for insertions and predefined directories for extensions, which are both stored (“harvested”). This allows the harvested code changes to be injected in the newly expanded code base, and extended files to be added in the appropriate

directories. The harvesting mechanism therefore leads to a clean separation between the expanded code base and its customizations. Given the recurrent structure of the expanded code base, the main complexity of the application becomes determined by the customizations, rather than the expanded code base itself. In PEMM v2, customizations only represent a small fraction of the overall code base: 5 percent.

Evolvability: The harvesting/injection mechanism enabled new dimensions of evolvability: as customizations could be applied to a newly generated code base, both could start to evolve independently. The obsoleteness of marginal expansions illustrates the usefulness of the mechanism. Additional data attributes could now be simply added to the descriptors, upon which a completely new code base (including injected customizations) could be generated. Similarly, a new code base could be generated based on new technology versions. For example, when a new version of the presentation framework provides new features which are included in the expanders, a new code base could be generated and injected with the customizations. The application is then enhanced with the new features, without requiring additional effort. As a result, the technologies used in the application could easily be kept up to date.

IV. Discussion

In this section, we respectively discuss some case reflections (Section IV-A), introduce a generalization of the four case phases (Section IV-B) and discuss the implications of the case findings for other enterprise layers (Section IV-C).

A. Case Discussion

Certain noteworthy reflections in relation to the current state-of-the-art in software engineering can be made based on the case documented in the previous section.

First, a tendency has been observed to deprecate or "throw away" large portions of code when functional requirements or team members change. This "*not invented here syndrome*" [14] typically results in a lot of rework and little reuse. An illustration of this phenomenon is the redevelopment of the application after phase 1. The case further illustrates how, in subsequent phases, this inclination can be mitigated by applying a fine-grained, reusable code structure. Many functional changes have been implemented, and different programmers have been working on the application, without the need to deprecate the existing code base. This reuse enabled a focus on applying functional changes, rather than reworking architectural aspects of the code. As a result, the application has been updated regularly during 7 years, without requiring large code deprecations.

Second, the contrast between the widely used design patterns and software elements as proposed by NS Theory should be noted. While design patterns might incorporate substantial design knowledge regarding evolvability, their concrete implementation is still left to the programmer. Applying multiple design patterns simultaneously results in a complex structure, which, if created by hand, is error-prone and difficult to maintain. This has been acknowledged by various scholars reviewing the state-of-the-art of design patterns: "*general design principles can guide us, but reality tends to force trade-offs between seemingly conflicting goals, such as flexibility and maintainability against size and complexity*" [15, p. 88].

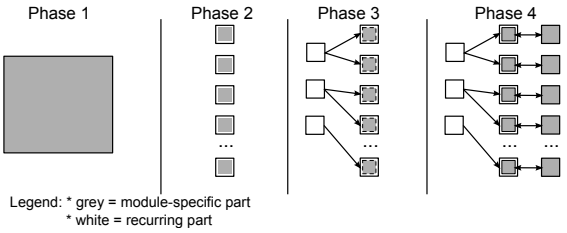


Figure 1. 4 phases, each one with their distinct variability dimensions.

Phase 2 of the case, where the code structure needed to be recreated manually, illustrates this. However, in subsequent phases, code reuse was enabled by the expansion mechanism. As a result, manual coding was no longer required to apply existing design knowledge, leading to a consistent and correct application of the accumulated knowledge. The mechanism of applying design knowledge through the use of expanders has been discussed in [16].

Third, the lack of an expansion mechanism jeopardizes true black-box reuse of software modules. As argued in [4, p.178]: "*in many cases, the problem is not that the component cannot be found in a repository, or cannot be reused at this point in time, the main problem is that this would create dependencies of which the future implications are highly uncertain.*" This implies that white-box inspection remains necessary in order to safely reuse modules in an evolving system, since not all dependencies of a module can be considered to be visible in its interface (i.e., hidden dependencies exist). The expansion mechanism allows a one-time inspection of a modular structure and, because of the systematic duplication of that structure, a deterministic construction (guaranteeing the absence of additional dependencies) of (re-)expanded code. Moreover, re-expansions, such as illustrated in PEMM v2, even allow the removal of newly discovered hidden dependencies in the elements within older NS-based applications.

B. Generalizing the four phases to modular structures

The four versions of the SMS/PEMM system as described in Section III can actually be analyzed in terms of general modularity structures as well. That is, in each of the four phases, a more NS-like approach was adopted in which the software code was modularized in a more fine-grained way. Reflecting on the essence of each of these phases may help us in applying NS reasoning to modular systems in general and to other application domains, such as modular organizational artifacts. Figure 1 provides a visual overview of these four phases in general modularity reasoning. We will now briefly discuss each of them separately, while paying specific attention to the *variability dimensions* and *recombination potential* in each of the phases. The former specifies the dimensions along which the evolution of the system takes place. The latter quantifies the number of different ways in which the system can be arranged, based on its modular structure.

1) Phase 1: one monolithic block.: In the first phase, the application could more or less be considered as one monolithic block. While any application itself obviously consists out of a set of modular primitives as provided by the programming language used (e.g., functions, structs, classes), no special attention was given to a purposeful delineation of parts within

the overall application. This is similar to a general system in which no deliberate modularization is introduced. Flexibility in such design is clearly limited as the *recombination potential* basically only equals 1. This means that no parts of the system can be re-used (within one system or between several systems) or separately adapted and combined with other parts. As a consequence, adaptations are mostly not contained into one well-defined part of the overall system. Therefore, the *variability dimension* is the system as a whole and Figure 1 therefore only represents one grey box for this phase.

2) Phase 2: identifying a first set of modules.: In the second phase, the application was more purposefully structured, i.e., different parts were deliberately separated (e.g., classes for local interfaces, home interfaces and agents). As the modularization was already quite fine-grained and similar functionality was required several times, a certain repetitiveness in the code base became clear. Therefore, in order to clarify this way of working in general modularity reasoning, the second column of Figure 1 first of all represents a system as consisting out of several subsystems or modules. Additionally, this phase already exhibits a special kind of modularity as each of these modules consists out of a manually constructed (mostly) recurring part (i.e., the white part) and a part specific for that module (i.e., the grey part). This means that the *variability dimension* is redirected towards the set of individual modules: the goal of modularity is that each of the modules can be adapted (e.g., upgraded to a newer version) and be plugged in into the system as a whole (i.e., recombined with one another). Based on such modular design, the *recombination potential* becomes k^N when having N modules with each k versions. Increasing N or k therefore significantly increases the recombination potential of such system.

However, it needs to be mentioned that subdividing a system and creating versions of each of these subsystems in order to increase the recombination potential is only successful if done wisely. That is, problems arise due to coupling at the *intramodular level* if content is duplicated among modules. Coupling at the *intermodular level* may arise if dependency rippling occurs. Regarding the former, applying a change to a duplicated part requires each of the modules (in which the duplicated part is embedded) to be adapted. Regarding the latter, ripple effects may cause that a change in one modules requires all other modules (using this first module) to change as well in order to be still able the use first module. The NS theorems formulated in general modularity reasoning can therefore be argued to focus on designing systems which eliminate both such intramodular coupling (e.g., via Separation of Concerns) and intermodular coupling (e.g., via Version Transparency) [9].

3) Phase 3: reusing modular structures in a systematic way.: As from the third phase, the software developers stopped creating the recurring part manually and started generating these recurring parts. Therefore, the left part of the third column of Figure 1 shows general and explicitly predefined parts which can be used as the "background" for modules. More specifically, predefined structures for three types of modules are provided in this example (e.g., in a software context: an empty data, action, and flow element). In the right part of the third column, the "background" of the module is combined with the module-specific part in order to arrive at a fully functional module. When analyzing the *recombination potential* in this case, one has to consider both the versions of the predefined modules (e.g., an ameliorated predefined structure to encapsulate processing functions), as well as the versions of the module-specific parts (e.g., an updated processing function with a new encryption algorithm). Therefore, in case we consider only one predefined module type j, the recombination potential becomes $l \times k^N$ when having l versions of the predefined module type j, N instantiated modules of type j, and k versions of each module-specific part. It should be noticed that, for each new version of a predefined modular structure applied to an already existing module, the module-specific part should again be incorporated into this modular "background" manually. Therefore, this recombination potential cannot fully be realized at this stage of modularization yet.

4) Phase 4: expanding elements and harvesting customizations.: In the last phase, depicted in the fourth column of Figure 1, the module-specific parts can be isolated and separately stored (i.e., harvested, as represented in the right side of the column) before a regeneration of the recurring predefined modular structures is performed. Therefore, the module-specific parts do not have to be incorporated manually in this modular "background" any longer. Instead, these parts are automatically injected into the general parts at predefined locations. This enables to achieve the mentioned *recombination potential* in the previous phase in reality. Stated otherwise, two different *variability dimensions* have to be considered. First, we have the different versions of the module-specific parts. Second, there are different versions of the module-generic parts. This means that classical version numbering in such cases becomes rather useless: it does not make a lot of sense anymore to consider a fixed "version" of the modular system as it is the result of the combination of two different variability dimensions (i.e., all general parts and module-specific parts can have different versions).

C. Implications for other enterprise layers

While the core of this paper discussed how NS Theory specifically modularizes the software layer within an enterprise architecture, the previous subsections reflected on the implications for the software engineering field and modular systems in general. Based on these reflections, some preliminary implications for other enterprise layers can be discussed.

First, the case illustrates how software evolvability was enabled by allowing changes to small modules, rather than updating one large, monolithic design (cfr. removing the need for code deprecation), as discussed in Section IV-A. In current enterprise architecture approaches, AS-IS and TO-BE versions are typically designed. Here, the focus is mainly directed to two separate, monolithic designs as opposed to gradual changes to small, individual modules. For truly evolvable enterprise architecture layers, the evolution of smaller modules should be addressed.

Second, the usage of repetitive structure instantiations through expanders was contrasted with the documentation of more generic design patterns (cfr. Section IV-A). On different organizational layers, recurring structures or patterns have been proposed as well [17]. Initial explorations of organizational patterns ("elements"), conform with NS Theory, have already been presented [3], [9]. For instance, De Bruyn [9] conceptually suggests a set of possible cross-cutting concerns and ele-

 ISBN: 978-1-61208-438-1

ments at this level. Currently, these approaches have however not yet been implemented in practice and should be further elaborated in future research. Ultimately, this would lead to the interesting phenomenon of clearly described variability dimensions within organizational layers, which provides decision makers with clear options for evolving the organization [18].

Third, we discussed how NS Theory demonstrates the difficulty of designing truly black-box modules (cfr. Section IV-A). It has been argued by various scholars that several other layers within organizations can be considered as modular structures as well [19]. Based on these arguments, several attempts have been made in the past to apply NS reasoning to such layers, such as business processes and enterprise architectures [8], [3], [9]. These efforts mainly concentrated on identifying and proving the existence of combinatorial effects in a diverse set of organizational layers and functional domains [8], [3], [20], [21], demonstrating a similar issue: it is hard to create truly black-box, fine-grained modules on these levels as well.

Fourth, the concept of recombination potential demonstrates the relevance of addressing the difficult research challenges outlined in the previous implications. Achieving a larger recombination potential in organizational artifacts such as products, processes and departments would (1) enable mass customization of products, which currently still results in high costs and a large complexity [22]; (2) provide a systematic approach to versioning artifacts, which is a large issue when implementing innovation at a steady pace [1]; and (3) aid in executing complex mergers and acquisitions, by considering organizational departments as modular options [18].

V. Conclusion

In this paper, we discussed a longitudinal case study of an NS application. We focused on how an increasingly fine-grained software structure enabled different types of evolvability. Such a description contributes to the NS knowledge base, since it illustrates the theoretical implications of NS Theory. Following this description, we generalized our findings towards generic modular structures. Since different enterprise architecture layers have been considered as modular structures before, we applied the resulting insights to other layers. This effort contributes to ongoing research in the Enterprise Engineering field, by integrating the current paper with previous research, and exploring future research challenges.

References

[1] A. Van de Ven and H. Angle, An Introduction to the Minnesota Innovation Research Program. New York, NY: Oxford University Press, 2000.

[2] J. W. Ross, P. Weill, and D. C. Robertson., Enterprise Architecture as Strategy – Creating a Foundation for Business Execution. Harvard Business School Press, Boston, MA, 2006.

[3] P. Huysmans, "On the feasibility of normalized enterprises: Applying normalized systems theory to the high-level design of enterprises," Ph.D. dissertation, University of Antwerp, 2011.

[4] H. Mannaert and J. Verelst, Normalized Systems—Re-creating Information Technology Based on Laws for Software Evolvability. Kermt, Belgium: Koppa, 2009.

[5] H. Mannaert, J. Verelst, and K. Ven, "Towards evolvable software architectures based on systems theoretic stability," Software: Practice and Experience, vol. 42, no. 1, January 2012, pp. 89–116. [Online]. Available: http://dx.doi.org/10.1002/spe.1051

[6] G. Oorts, P. Huysmans, P. De Bruyn, H. Mannaert, J. Verelst, and A. Oost, "Building evolvable software using normalized systems theory: A case study," in System Sciences (HICSS), 2014 47th Hawaii International Conference on, Jan 2014, pp. 4760–4769.

[7] G. Oorts, K. Ahmadpour, H. Mannaert, J. Verelst, and A. Oost, "Easily evolving software using normalized system theory - a case study," in Proceedings of ICSEA 2014 : The Ninth International Conference on Software Engineering Advances. Nice, France: ICSEA, 2014, pp. 322–327.

[8] D. Van Nuffel, "Towards designing modular and evolvable business processes," Ph.D. dissertation, University of Antwerp, 2011.

[9] P. De Bruyn, "Generalizing normalized systems theory: Towards a foundational theory for enterprise engineering," Ph.D. dissertation, University of Antwerp, 2014.

[10] H. Mannaert, J. Verelst, and K. Ven, "The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability," Science of Computer Programming, vol. 76, no. 12, 2011, pp. 1210–1222.

[11] P. Huysmans, G. Oorts, and P. De Bruyn, "Positioning the normalized systems theory in a design theory framework," in Proceedings of the Second International Symposium on Business Modeling and Sofware Design (BMSD), Geneva, Switzerland, July 4-6 2012, pp. 33–42.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[13] G. E. Krasner and S. T. Pope, "A cookbook for using the model-view controller user interface paradigm in smalltalk-80," J. Object Oriented Program., vol. 1, no. 3, Aug. 1988, pp. 26–49. [Online]. Available: http://dl.acm.org/citation.cfm?id=50757.50759

[14] R. Katz and T. J. Allen, "Investigating the not invented here (nih) syndrome: A look at the performance, tenure, and communication patterns of 50 r & d project groups," R&D Management, vol. 12, no. 1, 1982, pp. 7–20. [Online]. Available: http://dx.doi.org/10.1111/j.1467-9310.1982.tb00478.x

[15] G. Hohpe, R. Wirfs-Brock, J. W. Yoder, and O. Zimmermann, "Twenty years of patterns' impact," Software, IEEE, vol. 30, no. 6, Nov 2013, pp. 88–88.

[16] P. De Bruyn, P. Huysmans, G. Oorts, D. Van Nuffel, H. Mannaert, J. Verelst, and A. Oost, "Incorporating design knowledge into the software development process using normalized systems theory," International Journal On Advances in Software, vol. 6, no. 1 and 2, 2013, pp. 181–195.

[17] J. Dietz, Enterprise Ontology: Theory and Methodology. Springer, 2006.

[18] C. Y. Baldwin and K. Clark, "The option value of modularity in design," Harvard NOM Research Paper, vol. 3, no. 11, 2002.

[19] C. Campagnolo and A. Camuffo, "The concept of modularity within the management studies: a literature review," International Journal of Management Reviews, vol. 12, no. 3, 2010, pp. 259–283.

[20] J. Verelst, A. Silva, H. Mannaert, D. A. Ferreira, and P. Huysmans, "Identifying combinatorial effects in requirements engineering," in Advances in Enterprise Engineering VII, ser. Lecture Notes in Business Information Processing, H. Proper, D. Aveiro, and K. Gaaloul, Eds. Springer Berlin Heidelberg, 2013, vol. 146, pp. 88–102.

[21] E. Vanhoof, P. Huysmans, W. Aerts, and J. Verelst, "Evaluating accounting information systems that support multiple gaap reporting using normalized systems theory," in Advances in Enterprise Engineering VIII - Fourth Enterprise Engineering Working Conference (EEWC 2014), ser. Lecture Notes in Business Information Processing, D. Aveiro, J. Tribolet, and D. Gouveia, Eds., vol. 174. Springer, 2014, pp. 76–90.

[22] J. H. Gilmore and B. J. Pine II, "The four faces of mass customization." Harvard Business Review, vol. 75, no. 1, 1997, pp. 91 – 101.

 ISBN: 978-1-61208-438-1

Applying ISO 9126 Metrics to MDD Projects

Ricardo Muñoz-Riesle, Beatriz Marín
Facultad de Ingeniería
Universidad Diego Portales
Santiago, Chile
e-mail: rm.riesle@gmail.com; beatriz.marin@mail.udp.cl

Lidia López
Software and Service Engineering Group
Universitat Politècnica de Catalunya
Barcelona, Spain
e-mail: llopez@essi.upc.edu

***Abstract*— The Model Driven Development (MDD) paradigm uses conceptual models to automatically generate software products by means of model transformations. This paradigm is strongly positioned in industry due to the quickly time to market of software products. Nevertheless, quality evaluation of software products is needed in order to obtain suitable products. Currently, there are several quality models to be applied in software products but they are not specific for conceptual models used in MDD projects. For this reason, it is important to propose a set of metrics to ensure the quality of models used in MDD approaches in order to avoid error propagation and the high cost of correction of final software applications. This paper analyzes the characteristics and sub-characteristics defined in the ISO/IEC 9126 quality model in order to reveal their applicability to MDD conceptual models.**

Keywords-Quality Model; Model-Driven Development; Metrics; ISO 9126; Conceptual models

I. INTRODUCTION

Software production processes based on Model Driven Development (MDD) generate software products automatically or semi-automatically from conceptual models by means of model transformations [1][2]. To do this, well-defined modeling constructs, model-to-model transformations and model-to-code transformations are needed. Therefore, MDD approaches uses as input conceptual models and models transformations in order to generate the programming code of software products. This type of development is strongly positioned in the software development industry [3] due to the automatic generation of code, which speed the time to market and avoids error propagation and the high cost of correction of human errors in manual programming.

The MDD software process is supported by the Model Driven Architecture (MDA) [4] standard. MDA defines four abstraction levels for the models used to generate a software product that goes from the higher abstraction level to the lower abstraction level. These levels correspond to the computation independent model (CIM), the platform independent model (PIM), the platform specific model (PSM), and the implementation model (IM). Therefore, the conceptual models used by MDD approaches at any level become an essential resource in the process of software generation due to they are the main input for code generation. In other words, CIM models are used to generate PIM models, PIM models are used to generate PSM models, and PSM models are used to generate the IM model, which corresponds to the code in a specific programming language.

The quality evaluation of these conceptual models is of paramount importance since it allows an early verification of final software products. However, there is no standard defined to evaluate the quality of conceptual models used at MDD environments. The ISO 9126 standard [5] presents a set of characteristics and sub-characteristics that allows the evaluation of the quality of a software product by using different quality metrics proposed for each characteristic. However, these metrics are applied to measure artifacts obtained in later stages of software development cycles, increasing the cost of detecting and correcting defects.

We advocate that it is possible to apply the standard ISO 9126 to evaluate software products that have been developed under an MDD approach. Thus, this paper analyzes the ISO 9126 characteristics, sub-characteristics and their metrics in order to fit an MDD development process, and therefore, evaluate the quality of MDD projects using the metrics defined by ISO 9126. To do this, an exploratory study about the applicability of the defined metrics to conceptual models at different abstraction levels of MDD approaches has been performed. Thus, the main contribution of this work is the selected set of metrics that can be applied to the conceptual models that are specified at the different abstraction levels regarding MDA. This set of metrics composes, therefore, a quality model that allows an early evaluation of software generated in MDD environments.

This contribution is useful for both practitioners and researchers. Practitioners can use this set of metrics in order to evaluate early the quality of models used for the generation of their software products, aligning this evaluation with the standard ISO 9126. Researchers can use this set of metrics in order to integrate it to other quality evaluation techniques used at early phases in the software development cycle.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 presents an exploratory study about the ISO 9126 quality model in order to evaluate the applicability of the defined metrics at MDD projects. Section 4 presents the application of the set of selected metrics of the quality model to a case study. Finally, Section 5 presents an overall analysis and some conclusions from the results obtained.

II. BACKGROUND AND RELATED WORK

This section introduced the ISO 9126 standard in order to facilitate the understanding of later sections. Afterwards, a discussion about relevant related works is presented.

 ISBN: 978-1-61208-438-1

A. ISO/IEC 9126 standard

The ISO/IEC 9126 standard consists of four parts: the quality model [5], the external metrics [6], the internal metrics [7] and the metrics for quality in use [8]. The first part describes in detail six quality characteristics for software products (functionality, reliability, usability, efficiency, maintainability, and portability), and their corresponding sub-characteristics (see Figure 1).

Functionality	Reliability	Usability	Efficiency	Maintainability	Portability
Suitability	Maturity	Understandability	Time Behaviour	Analizability	Adaptability
Accuracy	Fault Tolerance	Learnability	Resource	Changeability	Installability
Interoperability	Recoverability	Operability	Utilization	Stability	Co-Existence
Security	Compliance	Attractiveness	Compliance	Testability	Replaceability
Compliance		Compliance		Compliance	Compliance

Figure 1. ISO 9126 characteristics and sub-characteristics.

In order to determine the level of quality of software products, it is necessary to evaluate these characteristics by applying a set of well-defined metrics. Thus, the second, third and fourth part of ISO 9126 describes metrics to assess the software quality. These metrics are focused on measuring artifacts obtained in later phases of the software development cycle, complicating the detection and correction of problems at early stages, which are propagated to later stages.

B. Related Work

The quality evaluation of software products is the paramount importance. For software products that are developed using an MDD approach, the quality evaluation can be performed at the conceptual models that are used as input for the automatic code generation.

There are several works that are focused in the quality evaluation of software products generated in an MDD approach. A mapping study of works that are focused in quality at Model-Driven Engineering (MDE) was presented in [9]. The main concerns presented in this study are (1) the studies do not provide an explicit definition of quality in model-driven contexts; (2) studies that are focused in UML models are not aligned with MDD approaches; and (3) there is a lack of analysis that indicates when a metric may or may not be applied to an MDD approach.

An approach to integrate usability evaluations of ISO 9126 into Model-Driven Web Development was presented at [10]. This study shows how the final user interface can provide feedback about changes in order to improve usability issues at intermediate artifacts of MDD projects (PIM and PSM models). This paper presents a similar way to us to analyze if the ISO 9126 metrics can fit into the MDD approach.

In [11], a quality model for MDD projects was presented. This model allows the verification of conceptual models by using a set of rules to detect defects related to data, process and interaction perspectives.

As summary, even though there are several model-driven proposals defined, and also there are several works that are1 focused on quality evaluation of MDD projects, there is a lack of a standard method to evaluate the quality at the different abstraction levels in model-driven approaches. For this reason, we decided to analyze the overall software quality framework presented by the ISO 9126 in order to identify if the metrics presented can be applied to the conceptual models defined at the different abstraction levels of MDD approaches.

III. EXPLORATORY ANALYSIS OF ISO 9126

This section presents an analysis of each characteristic of ISO 9126 and their corresponding sub-characteristics, considering if they are or not suitable to be applied to software products developed by using an MDD approach. In addition, the abstraction level of each presented metric has been identified.

A. Functionality

Functionality has been defined as the capability of a software product to provide the functions that meet the stated and implied needs when the software is used under specific conditions [5]. Requirements specifications are used to define the functions that meet the needs of users. Thus, to evaluate functionality it is necessary to focus in this kind of specifications.

In MDD projects, the requirements specifications can be performed by using CIM models, for instance i* models [12] [13], use-case diagrams [14], or BPMN diagrams [15]. Later, these models are transformed into PIM models (such as class diagrams) in order to continue with the process of code generation by using an MDD approach. These transformations are performed by different MDD approaches with their supporting tools, such as [16][17]. Thus, it is possible to evaluate the functionality of a software product generated in an MDD environment by using the CIM models that correspond to the requirements models. In other words, it is possible to evaluate if the software provide the functions stated in requirements by using CIM models and the traceability [18] of these models to the final programming code.

Functionality is comprised of the following sub-characteristics: suitability, accuracy, interoperability, security and conformance. For all the metrics, the closer to 1, the better:

1) Suitability: It corresponds to the capability to provide an appropriate set of functions for specific objectives [5]. In this context, the appropriateness is understood as the ability to correctly select a set of functions that meet the user needs. This verification process can be performed comparing the CIM with the PIM, or the CIM with the IM generated. The following metrics has been defined to evaluate the suitability [6]:

- Functional Adequacy (FA): This metric evaluates how adequate are the functions by using the formula presented in (1). This metric is useful for MDD approaches since it can be used at CIM or PIM. To do this, it is necessary to apply inspection techniques to find problems in the functions specified at CIM or PIM. For instance, if an MDD approach uses a class diagram as PIM, it is possible to identify defects in the defined functions by using for example a list of possible defects [19][20].

FA = 1-(Number of functions in which problems are detected/Number of functions evaluated) (1)

- Functional Implementation Completeness (FIC): This metric evaluates how complete is the implementation according to requirement specifications using the

formula presented in (2). This metric allows the identification of missing functions, based on the requirement specifications. Note that in MDD approaches the requirements are specified at CIM. Thus, the verification process can be performed by evaluating if the functions specified at CIM are present at PIM after the transformation from CIM to PIM, and also, it can be performed by evaluating if the functions specified at PIM are present at IM after the transformation process. For example, if an MDD approach uses i* models as CIM, it is possible to apply rules to verify that all the elements specified in the i* models are in the class diagram by using [21].

FIC = 1-(Number of missing functions detected/Number of functions described in Req Spec) (2)

- Functional Implementation Coverage (FICo): This metric evaluates how correct is the functional implementation using the formula presented in (3). This metric identifies those functions that have been incorrectly implemented or have not been implemented instead they have been specified in requirements models. In MDD projects, the code (IM) is automatically generated by the compilers, so that, to use this metric it is necessary to evaluate the CIM and the IM. Note that to go from CIM to IM it is necessary to go from CIM to PIM, then from PIM to PSM, and later for PSM to IM.

FICo = 1-(Number of incorrectly implemented or missing functions/Number of functions described in Req Spec) (3)

2) Accuracy: Corresponds to the capability of the software product to provide the right or agreed results or effects [5]. In order to measure the accuracy, it is necessary to define the concepts of trueness and precision. Trueness refers to the closeness of the mean of the measurement results to the true value; and precision refers to the closeness of agreement within individual measurement results. Therefore, according to the ISO standard, the term accuracy refers to both trueness and precision [22].

In order to measure the accuracy, two metrics has been defined [6]: Accuracy to expectation (AE) and Computational accuracy (CA). AE evaluates the actual results against the reasonable expected results in the operation time. CA evaluates how often the user found inaccurate results during the operation time. In both cases, to evaluate the accuracy is necessary to have the user executing the code. Thus, these metrics are used in later phases of the software development, so that, they do not contribute to the early quality evaluation of MDD projects.

3) Interoperability: Corresponds to the capability of a software product to interact with one or more specified systems [5]. The interoperability of a software product can be specified at the conceptual models that are used as input in an MDD approach. To do this, interfaces with other systems must be defined in the conceptual model to specify the data inputs and outputs.

The following metric is defined to evaluate the interoperability in [6]:

- Data Exchangeability (DE): This metric evaluates how correctly have been specified the exchange functions for specific data transfer using the formula presented in (4). To evaluate this metric it is necessary to specify the data formats that are exchanged with other systems and then apply inspection techniques to verify that the functions defined to exchange data are correctly defined. This metric can be evaluated at PIM of MDD approaches, where it is possible to specify the interaction with other systems. For instance, the MDD OO-method approach [23] allows the specification of *Legacy Views*, which corresponds to the abstraction of a software component that is represented by a class. The specification of the attributes and services of a legacy view requires the characterization of the functions or procedures that effectively carry out the corresponding function at other systems. By doing this, it is possible to identify whether software functions are compatible with the software that is specified by using the MDD approach. In addition, if the other system it is also specified by using an MDD approach, then, it is possible to specify the interactions between both systems at the metamodel level [24]. Thus, to verify this metric it is necessary to inspect the models following the syntax, semantics and restrictions specified in the metamodels.

DE = (Number of data formats which are approved to be exchanged successfully with other software or system during testing on data exchanges/Total number of data formats to be exchanged) (4)

4) Security: Corresponds to the capability of the software product to protect information in order to avoid unauthorized people or systems to read or modify them; and to provide authorized people or systems to have access to them [5]. The MDD approaches allow the specification of the users of the generated software. For instance, the OO-Method MDD approach allows the specification of agents at PIM, which have access to perform specific tasks and to read specific data.

The following metrics have been defined to evaluate the security [6]: Access Auditability (AA), Access Controllability (AC), and Data Corruption Prevention (DCP). AA, AC and DCP are calculated by using information of the user access at the operation time. Therefore, they do not contribute to the early evaluation of the quality of MDD projects. Nevertheless, it is important to note that in MDD projects the code is automatically generated from an input conceptual model. Thus, if an erroneous access is found in the access to the functionality, it can be corrected at PIM, and then regenerate the code.

B. Reliability

Reliability corresponds to the capability of the software product to maintain a specified level of performance when it is used under specified conditions [5]. To evaluate the performance it is necessary to execute the software, so that it cannot be simulated at design time of software, and it is

 ISBN: 978-1-61208-438-1

necessary to use the IM model, which corresponds to the code automatically generated by the MDD approach.

C. Usability

Usability corresponds to the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use [5]. This characteristic and its sub-characteristics are usually used once the software is executed, but we advocate that it is possible to measure this characteristic at early stages of the software development cycle by using an MDD approach. To do this, it is necessary that the conceptual model of the MDD approach has a holistic representation of the system, i.e., including the specification of the structure of the system, the behavior of the system, and the graphical user interface.

In [25], a new sub-characteristic of usability is presented: Complexity, which can be applied to MDD approaches. Two metrics have been defined to evaluate the complexity in the use of interfaces and operations in software.

- Complexity: This metric provides an indicator that measures the average number of operations per offered interface [25] using the formula presented in (5). For MDD projects, this metric can be evaluated by using the specification of the graphical user interfaces defined in the presentation model and the services that are accessed from these interfaces. Thus, this metric can be applied at the PIM of MDD approaches. This parameter can be compared with the user's opinion on how hard it is to use all the operations of a specific interface. This would indicate the perceived level of complexity if the system has high complexity or low complexity by using the IM model in order to define the acceptable value of this metric.
- Interface Defects Avoidance (IEA): This metric defines the level of understanding of a user after a defect occurs. The closer to 1, the better. IEA uses the average number of graphic operations failed recognized by the user in comparison to the total defects pre-defined by the developers. Thus, this metric is evaluated when the software is executed, so that it does not contribute to the early quality evaluation of MDD projects.

Complexity = (Operations in all offered interfaces/Offered interfaces) (5)

D. Efficiency

Efficiency corresponds to the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions [5].

Unfortunately, there are many external factors, such as bandwidth, hardware, and number of users connected, which cannot be known at early stages of the software development cycle since they cannot be specified in the conceptual model. These factors are only known when the software is executed, so that this characteristic cannot contribute to the early evaluation of quality of MDD projects.

E. Maintainability

Maintainability corresponds to the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software, and also, in requirements and functional specifications [5].

The maintainability can be evaluated in the conceptual models used by MDD approaches. An MDD approach allows the automatic generation of code by using as input a conceptual model, thus facilitating the detection of defects in the final product, and the corresponding corrections at the conceptual model. In addition, the automatic code generation allows software analysts to easily return to the initial steps of the software development cycle in order to include improvements or adaptations in the conceptual model. For this reason, the sub-characteristics of maintainability are also analyzed.

1) Analyzability: Corresponds to the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified [5]. The metrics defined in [6] are focus to measure analyzability by observing the user's behavior, so that they do not make a contribution as a quality metric for MDD approaches.

2) Changeability: Corresponds to the capability of the software product to enable a specified modification to be implemented [5]. One of the main advantages of MDD approaches is the ease of change. This is due to the great advantage of the automatic generation of code that allows the quick return to any stage of the development cycle, and therefore, correct the problem by redefining the models of the software.

The metrics defined in [6] are not useful to define a quality model for MDD approaches regarding the changeability, because these metrics are focused on the user behavior using the software at a specific time, instead of measuring the behavior of the software itself. Despite this, we found a metric in [25], which has been defined to evaluate the changeability:

- Customizability Ratio (CR): This metric provides an indicator of the ability of modification of the software using the formula presented in (6). If the software offers few interfaces and many parameters, normally it would be very modifiable, though difficult to handle, while one with many interfaces and few parameters is slightly modified. This metric can be evaluated by using the PIM of an MDD approach.

CR = (Number of parameters/Number of interfaces offered) (6)

3) Stability: Corresponds to the capability of the software product to minimize unexpected effects from modifications of the software [5]. In [6], there are defined metrics focused on user's behavior so that they do not perform a contribution for the early quality evaluation of MDD.

4) Testability: Corresponds to the capability of the software product to enable modified software to be validated [5]. Software developed by MDD approaches can be easily tested by the automatic generation of code and test cases [26]. This allows testing the software model based on the software

 ISBN: 978-1-61208-438-1

requirements specification. If a problem occurs, it can be solved by returning to the initial stages of software development cycle.

For this reason, the metrics defined in [6], do not contribute to the early quality evaluation of MDD projects because they are focused on user's behavior.

F. Portability

Portability corresponds to the capability of a software product to be transferred from one environment to another [5]. In MDD approaches, the software products are developed under specific requirements, using models such as PSM [3]. Therefore, the same conceptual model can be used on different platforms assuring portability. Unfortunately, metrics defined in [6] doesn't help to evaluate the quality for MDD approach. This is because the metrics defined by the ISO 9126 are focused on reuse of the software developed. In contrast, MDD approaches allow going one step back and re-compile the conceptual model to different technological platforms by using different PSM and compilers.

G. Other Metrics of ISO 9126

The metrics presented with their formula are focused on measuring quality for MDD approach at an early stage of software development. Nevertheless, there are other metrics defined in the ISO 9126 standard that cannot be used to measure the quality of models used at MDD approaches, since they are used in final stages of software development, i.e., they need the execution of the software to be tested or they are focused on the user's behavior.

These metrics are (1) Functional Specification Stability (FSS), which counts the number of functions changed after entering in operation; (2). Precision (P), which counts the number of results with a level of precision different from required during the operation time; (3) Data exchangeability by the user (DEu), which counts the number of cases in which user failed to exchange data with other software or systems.

IV. CASE STUDY

This Section exemplifies how the metrics are used at a software development project using an MDD approach. To do this, we present a system called SICOVE, which corresponds to a vehicle trading system that supports the process of managing vehicles, premises, revenues and costs undertaken by a buy-sell generic vehicle company (accounting and taxes processes associated are excluded). Figure 2 shows the conceptual model for SICOVE system.

This conceptual model has been specified using OO-Method approach and the Integranova [27] tool, which is able to compile the conceptual model and automatically returns the generated code compiled to different platforms. To do this, the OO-Method conceptual model is comprised of four complementary views: the static view, the functional view, the dynamic view and the presentation view. The static view is specified in a class diagram, which allows the specification of the structure of the final system. The functional view is specified in a functional model, which allows the specification of the change of values of the attributes when a service is executed. The dynamic view is specified in a state transition diagram, which allows the specification of the valid lives of an object. The presentation view is specified in a presentation diagram, which allows the specification of the graphical user interface. We have selected this tool to apply the set of metrics since it is an MDD tool that has more than 10 years of successful usage in industry.

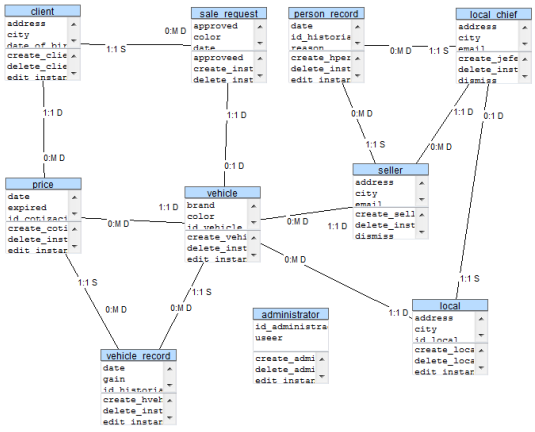


Figure 2. SICOVE Conceptual Model

Figure 2 shows the structural view of SICOVE system. All the functions of SICOVE have been specified by using the functional model (e.g., see Figure 3, which presents the specification of *create_client*). In Figure 3 it is possible to see the inbound arguments and the data type of each argument of the service.

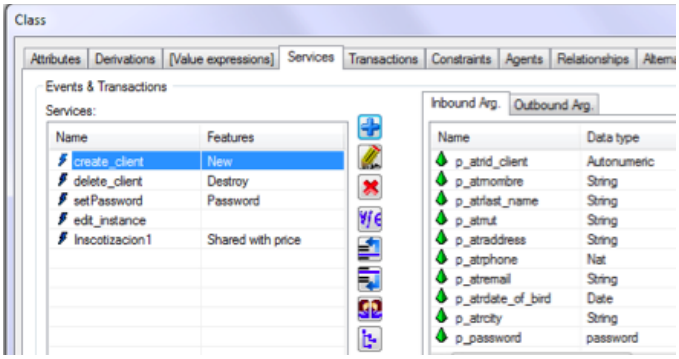


Figure 3. Example of SICOVE functional view

The generated code allows the testing of some of the functions of SICOVE system, for example create_client. In order to create a client we need to enter the following data into the system: id_client, name, last_name, rut, address, phone, email, date of birth, city (e.g., see Figure 4, which presents the attributes for the class client). Once entered the data, the system verifies that the user is not registered in the database in order to add it.

The SICOVE system has been used to evaluate the applicability of the metrics proposed in Section III. To do this, an analysis of all the functions defined in the specification of the system was performed in order to evaluate each metric proposed for MDD.

Table I shows the results obtained by applying the proposed metrics to the SICOVE system. This data was calculated by using the mathematical formulas described before, the requirements specification of the vehicle trading system that

 ISBN: 978-1-61208-438-1

was performed by using the IEEE 830 standard, and the conceptual model defined by the OO-Method approach, which correspond to the PIM abstraction level of MDA.

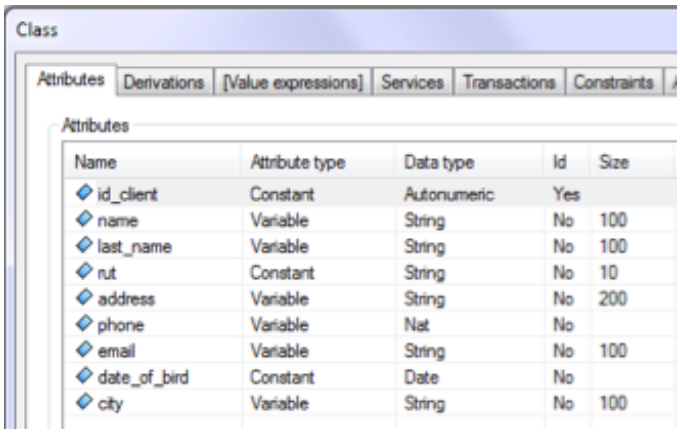


Figure 4. Example of SICOVE attributes

Regarding the functionality, FA, FIC and FICO metrics obtain a value less than one. This means that there are some functions that have been specified in the requirements but they do not have been generated at PIM. A summary of the functions defined in the requirements specification are presented in Table II. As this table shows, there is some functionality that is not fully present at the PIM conceptual model, such as Generate Quotation, Set Vehicle for Sale and Sell Vehicle. Thus, from a total of 17 functions defined for the SICOVE system, 3 of them are not fully implemented (e.g., see Figure 5). The result obtained after applying the mathematical formulas is 0.8 for each metric, which indicates that are some functions of the SICOVE system are not implemented. If the value of these metrics had been 1 this would indicate that all functions were correctly implemented.

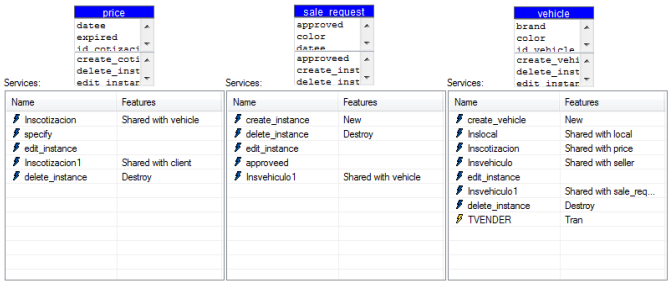


Figure 5. Functions specified for SICOVE system

TABLE I. RESULTS

Characteristic	Sub- Characteristic	Metric	Result
Functionality	*Suitability*	*FA*	*0.8*
		FIC	*0.8*
		FICo	*0.8*
	Interoperability	*DE*	-
Usability	*Complexity*	*Complex*	*5.3*
Maintainability	*Changeability*	*CR*	*7.1*

TABLE II. FUNCTIONS FOR SICOVE SYSTEM

ID	Functions of SICOVE system	Defined Functions	ID	Functions of SICOVE system	Defined Functions
1	Login to the system	Yes	10	Sell vehicle	**No**
2	Create user	Yes	11	View vehicle history	Yes
3	Edit user	Yes	12	View user history	Yes
4	Remove user	Yes	13	See income	Yes
5	Add local	Yes	14	View users	Yes
6	Modify local	Yes	15	View all local	Yes
7	Remove Local	Yes	16	View all vehicles	Yes
8	Set vehicle for sale	**No**	17	Generate quotation	**No**
9	Modify vehicle in system	Yes			

For DE metric, the SICOVE system works without requires inputs from other system. This means that is not dependant on other systems to perform their functions, so the connection between the SICOVE and other systems is not applicable. Thus, it is not possible to apply this metric in this case study.

For Complex and CR metrics, we identified 10 interfaces offered by SICOVE, 53 operations, and 71 parameters on all the graphical user interfaces offered, giving a result of 5.3 for Complex and 7.1 for CR metrics. In addition, the presentation view has been specified (e.g., see Figure 6, which presents the patterns to create the graphical user interface of new client service). The services with the arguments owned specified in Figure 3 are specified in Figure 5 as service interaction units. These results give us an indication of the current status of Complex and CR of the SICOVE system. These results indicate a normal level of complexity and customizability ratio to this system, due to it has the basic functions and parameters for the system to work.

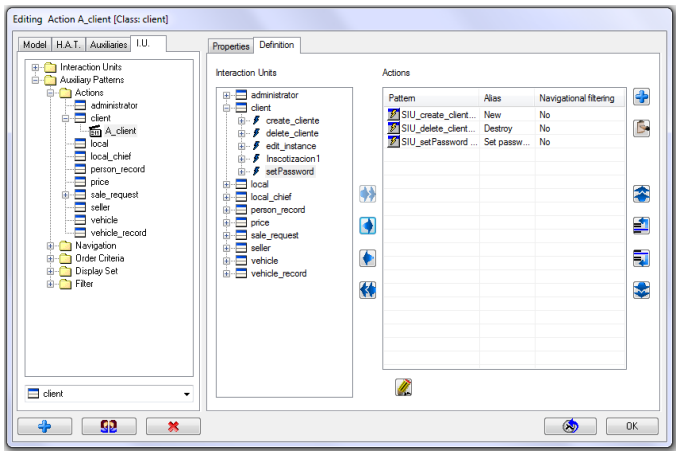


Figure 6. Example of SICOVE presentation view

All these metrics were applied manually to study the SICOVE system, which was automatically generated by the Integranova tool. Even though the case is small, and consequently the data delivered by not too big, it is enough to

 ISBN: 978-1-61208-438-1

understand the applicability of the ISO 9126 metrics to a particular MDD project. Thus, in this section we have exemplified the application of ISO 916 metrics to an MDD project, so that we verify the applicability of the selected metrics of ISO 9126 to an MDD project.

V. CONCLUSIONS AND FURTHER WORK

Software quality involves a strategy towards the production of software that ensures the user satisfaction, the absence of defects, the compliance with the budget and time constraints, and the application of standards and best practices for the software development. Thus, different techniques can be applied to the different artifacts used along the software development process. The ISO 9126 standard is a well-known quality model for software systems, so that in this paper we present an analysis of ISO 9126 regarding their applicability to MDD projects.

In particular, this paper presents an exploratory analysis of the ISO 9126 metrics that was performed in order to identify the metrics that could be used at early stages of software development cycle by analyzing the abstraction level of the conceptual models at which these metrics can be used. These early stages correspond to the specification of conceptual models for the analysis and design of software systems. In MDD projects, these conceptual models are located at different abstraction levels, which are the CIM, PIM, PSM or IM. In addition, these metrics have been used in an MDD project in order to evaluate their applicability. To calculate these metrics, the conceptual models of an industrial MDD approach were used. So that, we can conclude that these ISO 9126 metrics allow the early evaluation of quality of MDD projects, i.e., these metrics are useful for MDD projects.

Nevertheless, in MDD approaches there are many edges where is still possible to make a contribution to improve the quality evaluation of MDD projects, for instance extending the analysis to modeling languages, modeling tools, and modeling transformations, i.e., evaluating the quality of projects generated in MDE environments. Thus, immediate future work considers the inclusion of other metrics in order to have a well-defined set of metrics that conforms the basis of a quality model for MDD. And, later, further work considers the quality evaluation of MDE projects. We are aware that additional evaluation of our proposal to real development scenarios is necessary. Therefore, we consider as future work the development of empirical studies to evaluate the effectiveness and benefits of using these metrics under MDD approaches in real MDD projects.

ACKNOWLEDGMENT

This work was funded by Universidad Diego Portales and the FONDECYT project TESTMODE (Ref. 11121395, 2012-2015).

REFERENCES

[1] O. Pastor, J. Gómez, E. Insfrán, and V. Pelechano, "The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming", Information Systems, vol. 26, 2001, pp. 507–534.

[2] B. Selic, "The Pragmatics of Model-Driven Development", IEEE Software, vol. 20, 2003, pp. 19–25.

[3] OMG. MDA Products and Companies. Available: [retrieved: October, 2015] http://www.omg.org/mda/committed-products.htm

[4] OMG, "MDA Guide Version 1.0.1", 2003.

[5] ISO/IEC, "ISO/IEC 9126-1, Software Eng. – Product Quality – Part 1: Quality model", 2001.

[6] ISO/IEC, "ISO/IEC 9126-2, Soft. Eng. – Product Quality – Part 2: External metrics", 2003.

[7] ISO/IEC, "ISO/IEC 9126-3, Soft. Eng. – Product Quality – Part 3: Internal metrics", 2003.

[8] ISO/IEC, "ISO/IEC 9126-4, Soft. Eng. – Prod. Qual. – Part 4: Quality-in-Use metrics", 2004.

[9] F.D. Giraldo, S. Espana, and O. Pastor, "Analysing the concept of quality in model driven engineering literature: A systematic review". IEEE Eighth International Conference on Research Challenges in Information Science (RCIS), 2014, pp 1–12.

[10] A. Fernandez, E. Insfran, and S. Abrahão, "Towards a Usability Evaluation Process for Model-Driven Web Development", I-USED'09, Uppsala, Sweden, 2009, pp.1-6.

[11] B. Marín, G. Giachetti, O. Pastor, and A. Abran, "A Quality Model for Conceptual Models of MDD Environments", Advances in Software Engineering, vol. 2010 - Article ID 307391, 2010, pp. 1-17.

[12] S. Abdulhadi, "i* Guide version 3.0", 2007.

[13] i*. Wiki Web Page. Available: [retrieved: October, 2015] http://istar.rwth-aachen.de/

[14] OMG, "Unified Modeling Language (UML) 2.4.1 Superstructure Specification " 2011.

[15] OMG, "Business Process Model and Notation (BPMN) 2.0", 2011-01-03 2011.

[16] M. Kardoš and M. Drozdová, "Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA)", Journal of Information and Organizational Sciences, vol. 34, 2010, pp. 89-99.

[17] B. Brahim, E. B. Omar, and G. Taoufiq, "A methodology for CIM modelling and its transformation to PIM", Journal of Information Engineering and Applications, vol. 3, 2013, pp. 1-21.

[18] M. Ruiz, Ó. P. López, and S. E. Cubillo, "A Traceability-based Method to Support Conceptual Model Evolution", CEUR-WS.org, 2014, pp-1-8.

[19] B. Marín, G. Giachetti, O. Pastor, "Applying a Functional Size Measurement Procedure for Defect Detection in MDD Environments" 16th European Conference EUROSPI 2009, Vol. CCIS 42, Springer-Verlag, 2009, pp. 57-68

[20] B. Marín, G. Giachetti, O. Pastor, and T. E. J. Vos, "A Tool for Automatic Defect Detection in Models used in Model-Driven Engineering", 7th International Conference on the Quality of Information and Communications Technology (QUATIC), Oporto, Portugal, 2010, pp. 242-247.

[21] G. Giachetti, B. Marin, and X. Franch, "Using Measures for Verifying and Improving Requirement Models in MDD Processes", 14th International Conference on Quality Software (QSIC), 2014, pp. 164-173.

[22] ISO, "ISO 5725-2 – Accuracy (trueness and precision) of Measurements Methods and Results – Part 2: Basic Method for the Determination of the Repeatability and Reproducibility of a Standard Measurement Method", 1994.

[23] O. Pastor and J. C. Molina, Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling, 1st edition ed. New York: Springer, 2007.

[24] O. Pastor, G. Giachetti, B. Marín, and F. Valverde, "Automating the Interoperability of Conceptual Models in Specific Development Domains", in Domain Engineering: Product Lines, Languages, and Conceptual Models, Springer, 2013, pp. 349-374.

[25] A. Mattsson, B. Lundell, B. Lings, and B. Fitzgerald, "Linking model-driven development and software architecture: a case study", IEEE Transactions on Software Engineering, vol. 35, 2009, pp. 83-93.

[26] P. Baker, Z. R. Dai, J. Grabowski, Ø. Haugen, I. Schieferdecker, and C. Williams, Model-Driven Testing: Using the UML Testing Profile: Springer-Verlag 2008.

[27] Integranova. (2015). Web Page. [retrieved: October, 2015] http://www.integranova.com

 ISBN: 978-1-61208-438-1

Evaluation of a Security Service Level Agreement

Chen-Yu Lee, Krishna M. Kavi,

Department of Computer Science and Engineering, University of North Texas
1155 Union Cir, Denton, TX 76203, United States
Email: cychrislee@ieee.org, Krishna.Kavi@unt.edu

***Abstract*—Data breaches are the most serious security breaks among all types of cybersecurity threats. While Cloud hosting services provide assurances against data loss, understanding the security service level agreements (SSLAs) and privacy policies offered by the service providers empowers consumers to assess risks and costs associated with migrating their information technology (IT) operations to the Cloud. We have developed ontologies to represent security SLAs so that consumers can understand cybersecurity threats, techniques for mitigating the risks, and their roles and responsibilities and those of the service provider in terms of protecting IT systems. Our ontological representation of security services offered by a provider allows the customer to evaluate the level of compliance with respect to federal regulations such as Health Insurance Portability and Accountability Act (HIPAA). In this paper, we also describe ways to quantitatively assess the strength of compliance and the quality of protections offered by an SSLA. We hope that our approach can lead to negotiated SSLAs.**

Keywords–service level agreement; SLA; security; SSLA; cloud computing.

I. INTRODUCTION

In 2014 and 2015, we have seen numerous and significant data breaches. In September 2014 Home Depot suffered a data breach of 56 million credit card numbers [1] and in October 2014, 1.16 million customer payment cards were stolen from Staples [2]. In February 2015, CareFirst BlueCross BlueShield announced that it was the target of a cyber attack that compromised the information of about 1.1 million current and former consumers [3]. Compromised information included consumer user names for CareFirsts website, as well as names, birth dates, email addresses and subscriber identification numbers. Most recently (June 2015), the US Office of Personnel Management revealed a data breach that lead to a foreign nation having access to millions of US federal employee records [4]. These incidents show that data breaches (or an unauthorized person gaining access to data) are the most prevalent types of security attacks. Some of these attacks involved very sophisticated techniques to circumvent several levels of cybersecurity protections.

The Federal Risk and Authorization Management Program (FedRAMP) is a government-wide program that describes a standardized approach to security assessment, authorization, and continuous monitoring of Cloud IT products and services. FedRAMP is the result of close collaborations among cybersecurity and cloud experts from the General Services Administration (GSA), the National Institute of Standards and Technology (NIST), the Department of Homeland Security (DHS), the Department of Defense (DOD), the National Security Agency (NSA), the Office of Management and Budget (OMB), and the Federal Chief Information Officers (CIO) Council. The assessment process is based on a standardized set of requirements in accordance with the Federal Information Security Management Act (FISMA). The NIST Special Publication (SP) 800-53 [5] controls security authorizations. NIST is also working on new guidance, SP 800-174, which will address the distribution and placement of security controls for cloud computing environments. The new guidance will list the controls needed for capabilities (or services), and displays how a cloud capability (or service) should be correctly and completely secured. Finally, the NIST Cloud Computing program plans to define security SLAs, security metrics, security intelligence and continuous monitoring based on previous documents SP 500-299 [6], SP 500-307 [7], SP 800-173, and SP 800-174. The Security Service Level Agreement (SSLA) can be used to improve the credibility and verifiability of security and privacy commitments made by cloud providers.

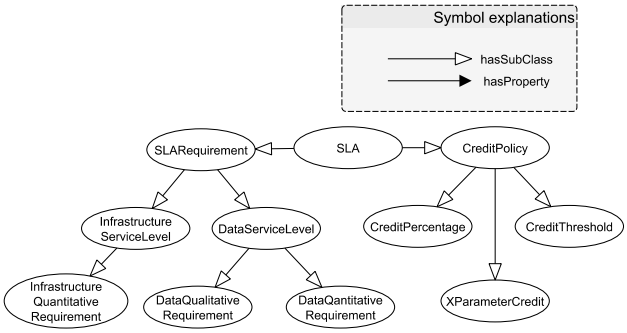


Figure 1. Ontology for SLA

In general, Service Level Agreements (SLAs) written by a Cloud provider are very difficult to understand, and it is even more challenging to quantitatively compare the SLAs of different providers. To capture and present requirements for both providers and consumers, Modica et al. proposed an SLA ontology that captures the definition of a semantic domain of knowledge for the cloud business (see Figure. 1) [8]. Based on the ontology knowledge base, providers can customize their offerings according to their business strategy, and consumers can request the resources and services consistent with their needs. However, this work does not cover security service levels, which led to our development of ontologies specifically for SSLAs.

This paper extends our previous work that proposed ontologies for SSLAs that could be used to understand the security

 ISBN: 978-1-61208-438-1

agreements of a provider and to audit the compliance of service levels with respect to federal regulations, such as HIPAA [9] [10]. We enrich the ontology models and propose an SSLA assessment system to evaluate the strength of agreements in terms of protecting IT assets. Our approach can be used to negotiate desired levels of security. The rest of the paper is organized as follows. Section II discusses research that is closely related to ours. The SSLA ontology framework is introduced in Section III. Our approach for assessing SSLAs is described in Section IV and we illustrate how this approach can be used for negotiating SSLAs in Section V. Section VI includes a discussion of our current research and our plans for extending the framework.

II. Related Works

A. Service Level Agreement

A SLA is a documented legal agreement between a service provider and a consumer and identifies services and levels of service targets based on the ISO 2000 standard for service management systems [11]. A Cloud Service level agreement is a document that states the services offered, performance levels and promises made by the cloud provider.

B. Security Service Level Agreement

The Security Service Level Agreement (SSLA) for specifying the security service requirements of an enterprise was first proposed by Henning [12]. Monahan et al. considered the issues of meaningful security SLAs and discussed how a security SLA embodies certain legal and contractual elements to satisfy two basic requirements: separation and compartmentalization [13]. In 2013, the terms SSLA and security service-oriented agreement were proposed by Takahashi et al. [14]. The authors proposed a non-repudiatable security service-oriented agreement mechanism that describes security requirements for users and capabilities of service providers. Rong et al. described some cloud security challenges including resource location, the multi-tenancy, authentication and trust of acquired information, system monitoring, and cloud standards [15]. Hale et al. built an XML-based compliance vocabulary compatible with the WSLA schema [16]. However, there are no prior attempts to describe SSLAs formally. Currently SSLAs are described informally in English, and it is very hard to evaluate or negotiate the strength of such informal descriptions. Previously we proposed an ontology for SSLA that covers the security issues required to meet most security regulations [9] [10]. This paper expands our previous ontology and proposes to evaluate the strength of an SSLA.

III. Ontology For SSLA

As an alternative to the traditional SLAs, written in natural languages, an XML-based SLA is more useful for automated processing. Previously we defined several different ontologies including ontologies for vulnerabilities, attacks and Security SLAs ([9] [10] [17]). Our ontology for Security Service Level Agreements (or SSLAs) are based on the design concepts of a trustworthiness ontology proposed in [18] and also extends Hale's work [16]. To increase the coverage of our SSLA ontology, we take into account the challenges in covering all control domains specified by the Cloud security alliance (CSA) Cloud Control Matrix (CCM) v3 [19] and the properties of some security frameworks, such as HITRUST [20].

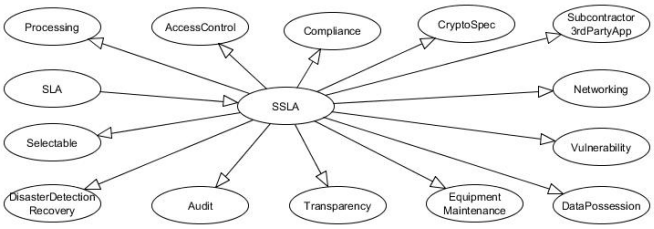


Figure 2. All classes in SSLA

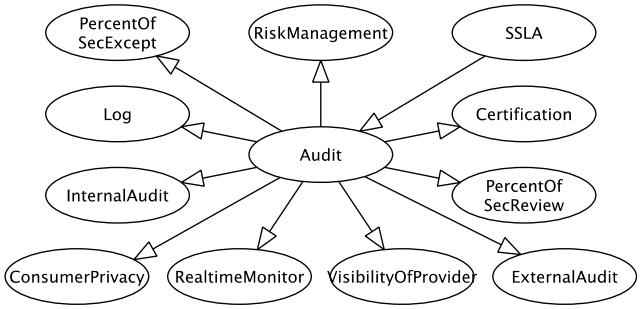


Figure 3. Audit class in SSLA

In this paper we extend our previous SSLA ontology so that SSLAs can be evaluated for their strengths. Our SSLA ontology facilitates an understanding of security concerns in service level agreements and allows one to match the security requirements of a customer with the SSLAs offered by service providers. Summarizing, our SSLAs offer these benefits.

- SLA agreements are easier to understand, particularly those related to security.
- During negotiations, consumers can compare the SLAs offered by different providers and choose the one that best fits their needs.
- It will be easier to monitor (or audit) the compliance of securities levels offerred by service providers with security requirements of federal regulations.

For completeness sake we introduce our SSLA ontology first. Without loss of generality, here we represent fourteen classes in our SSLA ontology, including `Networking`, `Vulnerability`, `Transparency`, `DisasterDetectionRecovery`, `DataPossession`, `CryptoSpec`, `AccessControl`, `Processing`, `Compliance`, `Audit`, `Selectable`, `Subcontractor3rdPartyApp`, and `Equipment Maintenance` as shown in Figure 2. Each class is described below. The ontology can be modified by removing or adding additional classes.

- Networking: This class organizes the agreements about the networking environment such as traffic isolation (`TrafficIsolation` subclass); IP and bandwidth monitoring (`BandwidthMonitoring` and `IPMonitoring` subclasses). These subclasses can be used to define functions such as allocating bandwidth and blacklisting (or whitelisting) IP addresses.

 ISBN: 978-1-61208-438-1

- Vulnerability: This class defines assurances in terms of detecting and patching known vulnerabilities. `PatchPolicyComplianceRate` and `ScanFrequency` subclasses can be used for specifying policies on how often the system is scanned for malware, and how soon a known patch is applied to remove vulnerabilities.
- Transparency: This class regulates the transparency of the information related to the security management processes used by the provider. The SSLA should record the responsible office that will provide the information regarding all security breaches and actions taken when requested.
- Disaster detection and recovery: This class describes the contingency plans and the security incident procedures, and details disaster detection and the recovery steps in the event of a breach. It may also define data backup functions because data is usually the most valuable asset for consumers.
- Data possession: This class controls data storage procedures and verification methods, and how often they are applied to ensure data authenticity. This class can be used to specify the ownership and the location of the storage.
- Audit: This class describes the processes for internal and external audits of the architecture, management, and services of providers, and the certificates obtained (listed in Certification) to build consumer trust in the providers as shown in Figure 3. `InternalAudit` and `ExternalAudit` subclasses also define the respective audit plans. `Log` is the most important evidence of behaviors of attackers, consumers, and providers. To protect the security of the log, the `Log` subclass regulates the secure storage and retention of the logs. The `RiskManagement` subclass describes the risk management and data risk assessment programs. The system administrators of the providers' systems have the highest level of privilege. They can perform any action on any object. Thus, the `ViabilityOfProvider` subclass defines what level of consumer data security is appropriate for a specific person and under what conditions. In addition, the class outlines the real time monitoring mechanisms, the acceptable percentage and types of security exceptions, security reviews, and the protection of consumer privacy in `RealtimeMonitor`, `PercentOfSecExcept`, `PercentOfSecReview` and `ConsumerPrivacy` subclasses.
- Subcontractor and third party application: This class clarifies the rights and duties with respect to security of the subcontractor and the third party application providers.
- Cryptography specification: Some providers offer encryption services. It is useful to optimize consumer data encryption while also reducing the associated computational complexity. Thus the level or type of encryption technique can be specified here.
- Access control: Access control of the instance control panel directly impacts the security of the instance. This class defines the access authentication, authorization, accounting schemes, including access using mobile devices. This class also can be used to specifiy the responsibility of the consumer in terms of permitting accesses within their user groups.
- Processing: This class covers the security demands for building a secure run time environment for virtual machine migration, queue service capability, virtual firewalls, isolation, portability and integrity of applications. Systems relying on hardware trusted platform modules may be viewed as providing higher levels of trust and this can be indicated in this class.
- Compliance: Some specific services must be certified as compliant with security and privacy standards, and practices as required by law. For example, user services that involve warehousing or mining of electronic Protected Health Information (ePHI), electronic Personally Identifiable Information (ePII), or Health Insurance Portability and Accountability Act (HIPAA) data must comply with all associated federal and local standards [21]. There are many subclasses defined in Compliance as shown in Figure 4. An SSLA can indicate the subclasses (or specific rules of the law) for which the provider is compliant.
- Equipment Maintenance: Keeping equipment maintained and upgraded may lead to fewer exploitable weaknesses. This class of our SSLA ontology defines the state of equipment, software versions and all upgrades since the installation.

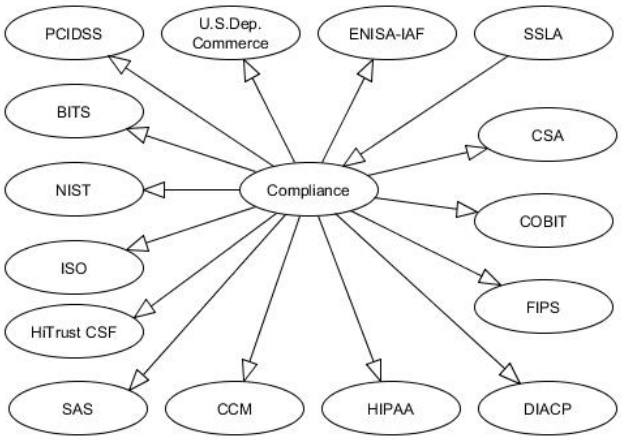


Figure 4. Compliance class in SSLA

IV. SSLA Assessment System

During the process of purchasing cloud services, a review of the service level agreement is the necessary phase where customers agree to a binding contract in term of the services received and payments made. At present, the agreement lists service guarantees and responsibilities of the provider. Often they are biased in favor of the provider; in many cases the customer is not afforded a chance to negotiate service levels. This is particularly the case when it comes to security levels offered by the provider. There is very little opportunity for the customer to explore whether the security is sufficient, or

if a lower or higher level security option is available. More importantly, the customer cannot evaluate the security levels using meaningful and quantitative measures.

Our SSLA ontology described in Section III contains fourteen classes and several subclasses that cover most of the security issues of interest. We feel that this allows one to map a SSLA contract to our ontology and evaluate the strength of security provided by the SSLA. In this section, we outline some potential ways for quantitatively assessing the strength of SSLAs.

A. Regulation Compliant

In general, a regulation describes rules, such as specifications, policies, standards, or law, especially the public regulations that apply in particular fields. Some examples of regulations include PCI-DSS [22], HIPAA and others shown in Fig.4. Each regulation defines different rules, but many rules in the regulations are similar. Therefore, an SSLA is stronger if it complies with more regulations.

B. Types of metrics

To evaluate an SLA, each individual (or a subclass) in our ontology has to be examined. Each entity should be quantified and we offer three different types of measurements for this purpose.

- Boolean measures (α): This type of quantification allows us to assess if a specific requirement (such as a specific HIPAA regulation or rule) is satisfied or not. Service providers will be fined if the provider fails to show that specific federal requirements are met. Note that different regulations (e.g., HIPAA, ENISA [23], PCI) may define different security requirements, and this translates to different subclasses (or individuals) in our ontology for meeting the requirements.
- Level measures (β): It should be possible to assess the strength of an agreement using qualitative measures as High, Medium, Low (or some other such levels). For example, in terms of the strength of encryption offered, one can say that using encryption algorithm Triple DES [24] is classified as low, but if one uses AES-128 [25] then the level may be viewed as medium, and the level is considered High if AES-192 or AES-256 are used for encryption. These are subjective assessments and we hope a consensus on the measurement can be reached through standards committees.
- Range measures (γ): These types of assessments can be used to define minimum threshold guarantees. For example, a user requires that the Cloud provider scan the systems for malware at least once every 12 hours. Any scanning rate below that can be viewed as less than satisfactory, and a value (say a percentage) may be assigned as a qualitative strength for the individual (or subclass).

C. Estimation of the security strength

We propose a quantitative analysis approach to estimate the security strength of each service level agreement. The process can follow the following outline.

Step 1: Prepare an ontology graph for the SSLA. Normally the ontology data can be stored in OWL or RDF format. The first step is to parse the ontology file as a graph for further query, e.g., RDFLib [26] in Python.

Step 2: Traverse all the individuals using SPARQL query. To examine each rule in the SSLA ontology, the approach traverses each individual with a recursive SPARQL query from the root through a class to each instance. SPARQL is a semantic query language for databases. It is used to retrieve and manipulate data stored in Resource Description Framework (RDF) format.

Step 2a: When visiting an individual (or a subclass), a score is assigned using the three types of measurement stated above.

$$P(\alpha) = \begin{cases} 1 & \text{if } \alpha \text{ is satisfied} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$P(\beta) = \begin{cases} score_{high} & \text{if } \beta \text{ is given a } \texttt{HIGH} \\ score_{medium} & \text{if } \beta \text{ is given a } \texttt{MEDIUM} \\ score_{low} & \text{if } \beta \text{ is given a } \texttt{LOW} \end{cases} \quad (2)$$

$$P(\gamma) = \{score_{range} \quad \text{if } \gamma \text{ is given } score_{range} \quad (3)$$

where $0 \leq score_{range}, score_{high}, score_{medium}, score_{low} \leq 1$ and the mapping scores from `HIGH`, `MEDIUM`, and `LOW` grades can be defined by the security committee.

Step 2b: The total score of a given SLA is $Score_{total}$.

$$Score_{total} = \sum_{i=1}^{n} class_i \quad (4)$$

$$class_i = \sum_{j=1}^{n} (P_j(\alpha) + P_j(\beta) + P_j(\gamma)) w_{i,j} \quad (5)$$

where $w_{i,j}$ is the weight of the j^{th} measurement of the i^{th} class and it can also be defined by security committee based on the emphasis level. The default value of $w_{i,j}$ is 1. Weights can be used to customize the measurements for individual needs. We describe the customization in the next section.

V. Customized Agreement

With our assessment system it is possible to compare SSLAs during the negotiation phase. An SSLA that scores highest is the optimal SSLA. This also means that the provider is held to very high levels of responsibility and liabilities, and this in turn can translate into higher cost to the customer. A customer should be able to understand the trade-offs between cost and the strength of an SSLA.

Figure 5 shows a comparison of two different types of companies. Figure 5(a) is a medical service provider that emphasizes compliance, access control, and audit classes of our ontology since these aspects of an SSLA are most important to their business. Other classes, such as networking or

encryption level are not as significant to their business (and does not interfere in demonstrating compliance with HIPAA regulations). On the other hand, Figure 5(b) is a company that offers online or downloadable games. Such a company is more interested in the security with on-line transactions (including payment transactions) and must be compliant with PCI DSS regulations. The company would also have significant interest in the access control, networking and infrastructure security. These examples are for illustration purposes only and the classes of companies used here are generic examples. More detailed analysis of each users requirements is needed to customize SSLA measurements. These two examples show that different types of companies may pay attention to the different classes of security needs. When negotiating SSLA, which part should be strengthened can be determined through the evaluation methods we describe in this paper. We assume that consumers will negotiate their customized SSLAs, instead of a generic SSLA offered by the provider. A generic SSLA may not be optimal in terms of cost and the level of security offerred. However, the generic SSLA may suffice for most customers.

VI. Discussion

This paper expands the SSLA ontology to cover more security regulations and security frameworks including HITRUST Cyber Security Framework (CSF). Therefore, in the next subsections, we first describe the implementation issue for the evaluation system for an SSLA based on the SSLA ontology. The system provides a quantitative result for the assessment that can be used to SSLA comparison and negotiation. Also, the coverage of HITRUST CSF is explained in subsection VI-B

A. Implementation

We implemented an SSLA assessment system to compute scores of the given agreement based on the approaches introduced in Section IV. Figure 6 is a snapshot of estimating HIPAA compliance in our assessment system. The program first shows each rule of the law for the consumer so that the consumer can understand the requirement. The quantitative scoring of the SSLA is based on the answers provided by the consumer. Current SSLAs are described in a natural language (i.e., English) and may be difficult to map onto our ontology. We require some input from the customer and service provider to interpret the SSLAs and map them to our ontologies. We hope future SSLAs will rely on more formal ontological definitions.

B. HITRUST Cyber Security Framework

The HITRUST Cyber Security Framework (CSF) is based on the Cyber Security Framework released by the National Institute of Standards and Technology (NIST) in February 2014. HITRUST CSF is a certifiable framework that provides organizations with a comprehensive, flexible and efficient approach to demonstrate regulatory compliance and risk management. Although HITRUST CSF is not a regulation, it provides for more security and privacy than HIPAA compliance. Figure 7 shows the fact that the HITRUSgrT properties subsume HIPAA rules related to access authorization. Thus if a provider satisfies the HITRUST CSF framework, the provider is also compliant with HIPAA regulations, as far as the access authorization class of our ontology is concerned. Likewise one can map the compliance with respect to other classes of the SSLA ontology.

C. Scoring system

The assessment system evaluates the SSLA quantitatively. In general, each mapped individual in our model is assigned one point; thus an SSLA with more points is assumed to be better as it satisfies more classes of our ontology. The weight valuable $w_{i,j}$ can be used to allow one to ignore some classes and place more emphasis on other classes.

D. Benfits of our Scoringe system

- For Cloud infrastructure provider: Since an ontology is a useful means for describing knowledge, a Cloud provider can employ our SSLA ontology to present the security levels guaranteed. Additionally, the SSLA ontology provides for negotiated agreements. With respect to HIPAA, the Cloud infrastructure provider must make sure that the Cloud environment is secure enough at least for known vulnerabilities and can resist known attacks. Moreover, the provider can use some vulnerability evaluation systems (like OKB [17]) to evaluate the security risks of its resources to define the most appropriate security guarantees, or price different levels of negotiated security agreements.
- For Cloud infrastructure users (primarily service providers): When service providers employ a Cloud environment, they can utilize our SSLA ontology framework to negotiate better levels of security guarantees from the infrastructure provider. Additionally, the service provider can use our framework to understand compliance issues about the services they offer.

VII. Conclusion

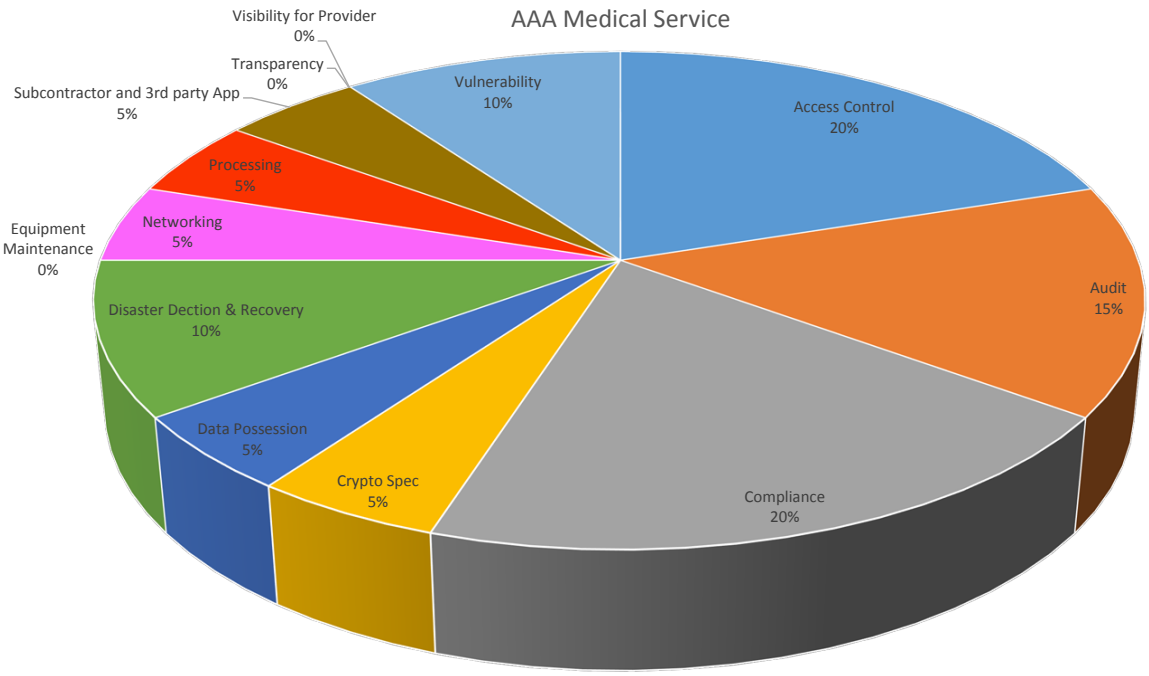
In this paper, we have developed an SSLA ontology framework that can be used to understand the security agreements of a provider and to audit the compliance of a provider with respect to federal regulations. The SSLA assessment system can be used to quantitatively measure the security strength of an SSLA, and can be used in the negotiation phase. In this paper, we are limited by the lack of accessible SSLAs of cloud providers such as Google, Amazon or Microsoft. We were only able to outline how HITRUST and HIPAA regulations translate into security requirements of individual IT systems and policies. It is our hope that the new federal guidelines and standards will force service providers to disclose details of their security SLAs. We will then be able to evaluate actual SSLAs of providers.

For future work, we plan to design SSLA templates for different types of industries with various levels of budgets based on the evaluation of collected agreements. These templates can be used to negotiate SSLAs with providers.

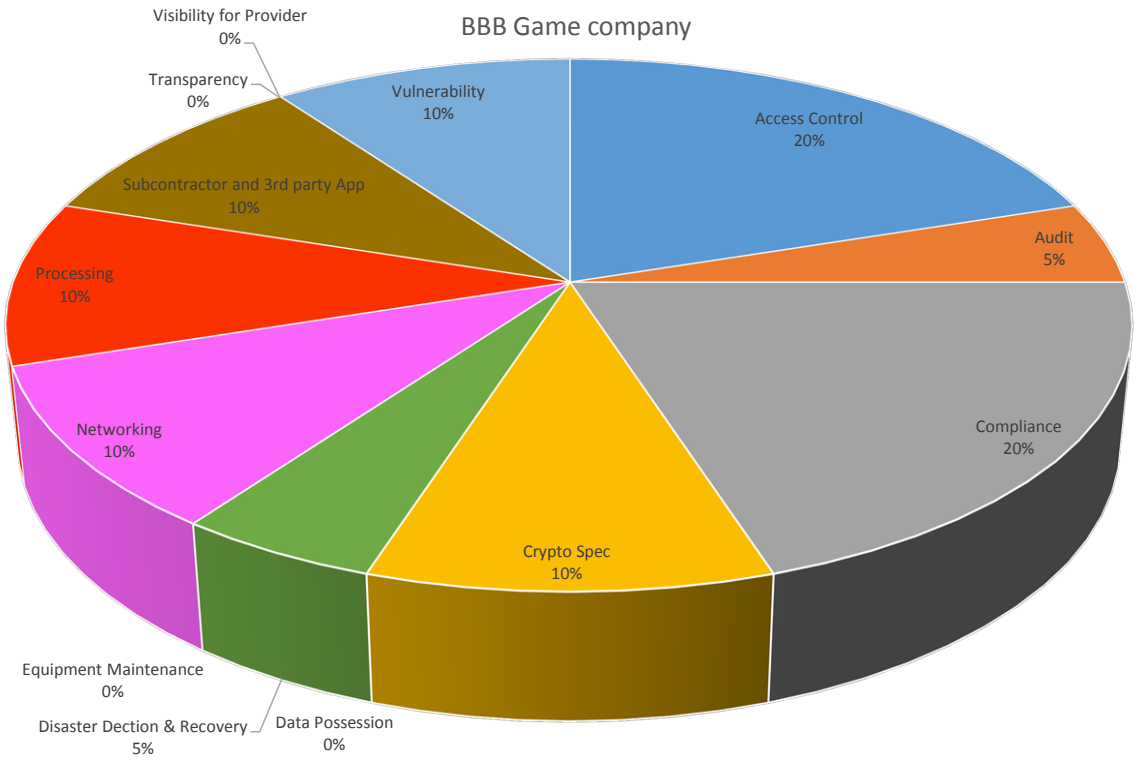
Acknowledgment

This research is supported in part by the NSF Net-centric and Cloud Software and Systems Industry/University Cooperative Research Center and NSF award 1128344. The authors want to thank David Struble for his editorial contributions to this paper.

 ISBN: 978-1-61208-438-1



(a) The proportion of the classes in a medical service's SSLA



(b) The proportion of the classes in a game company's SSLA

Figure 5. The proportion of the classes in SSLA

 ISBN: 978-1-61208-438-1

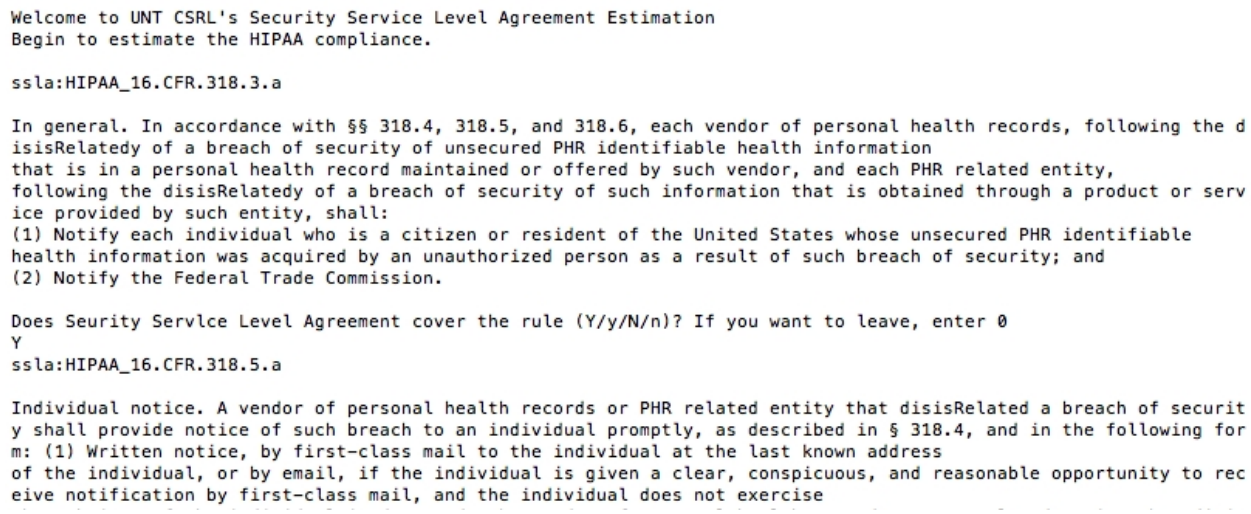


Figure 6. Snapshot of our SSLA assessment system. For estimating HIPAA compliance, the system first shows the rule of law, and the estimation is based on the administrators answer.

REFERENCES

[1] M. Backman, "Home depot: 56 million cards exposed in breach," Sep 2014, URL: http://money.cnn.com/2014/09/18/technology/security/home-depot-hack [accessed: 2015-09-15].

[2] J. Tom Huddleston, "Staples: Breach may have affected 1.16 million customers' cards," Dec 2014, URL: http://fortune.com/2014/12/19/staples-cards-affected-breach/ [accessed: 2015-09-15].

[3] D. Bowman, "Hack attack on carefirst compromises info for 1.1 million consumers," May 2015, URL: http://www.fiercehealthit.com/story/hack-attack-carefirst-compromises-info-11-million-consumers/2015-05-20 [accessed: 2015-09-15].

[4] "Office of personnel management data breach," June 2015, URL: http://www.c-span.org/video/?326593-1/hearing-office-personnel-management-data-breach [accessed: 2015-09-15].

[5] Security and Privacy Controls for Federal Information Systems and Organizations, SP 800-53, NIST, U.S. Department of Commerce Std., Rev. 4, Apr. 2013, URL: http://dx.doi.org/10.6028/NIST.SP.800-53r4 [accessed: 2015-09-15].

[6] NIST Cloud Computing Security Reference Architecture, SP 500-299 Draft, NIST, U.S. Department of Commerce Std., May 2013.

[7] Cloud Computing Service Metrics Description, SP 500-307 Draft, NIST, U.S. Department of Commerce Std., 2015, URL: http://dx.doi.org/10.6028/NIST.SP.307 [accessed: 2015-09-15].

[8] G. D. Modica, G. Petralia, and O. Tomarchio, "A business ontology to enable semantic matchmaking in open cloud markets," in Proc. SKG2012, Beijing, China, Oct. 2012, pp. 96–103.

[9] C. Y. Lee, P. Kamongi, K. M. Kavi, and M. Gomathisankaran, "Optimus: Framework of vulnerabilities, attacks, defenses and sla ontologies," International Journal of Next-Generation Computing, 2015.

[10] C. Y. Lee, K. M. Kavi, R. A. Paul, and M. Gomathisankaran, "Ontology of secure service level agreement," in Proc. HASE 2015, Jan 2015, pp. 166–172.

[11] Information technology. Service management. Service management system requirements, ISO/IEC 20000-1:2011, Std., 2011.

[12] R. R. Henning, "Security service level agreements: quantifiable security for the enterprise?" in Proc. NSPW 1999, Ontario, Canada, Sep. 1999, pp. 54–60.

[13] B. Monahan and M. Yearworth, "Meaningful security slas," HP Laboratories, Tech. Rep. HPL-2005-218R1, 2008.

[14] T. Takahashi and et al., "Tailored security: Building nonrepudiable security service-level agreements," IEEE VT Mag., vol. 8, no. 3, Sep. 2013, pp. 54–62.

[15] C. Rong, S. T. Nguyen, and M. G. Jaatun, "Beyond lightning: A survey on security challenges in cloud computing," Comput. Electr. Eng., vol. 39, no. 1, 2013, pp. 47–54.

[16] M. Hale and R. Gamble, "Building a compliance vocabulary to embed security controls in cloud slas," in Proc. SERVICES 2013, Jun. 2013, pp. 118–125.

[17] P. Kamongi, S. Kotikela, K. Kavi, M. Gomathisankaran, and A. Singhal, "Vulcan: Vulnerability assessment framework for cloud computing," in Proc. SERE 2013, 2013, pp. 218–226.

[18] R. Paul and et. al., "An ontology-based integrated assessment framework for high-assurance systems," in Proc. ICSC 2008, Aug 2008, pp. 386–393.

[19] "Cloud controls matrix version 3.0," Cloud Security Alliance.

[20] HITRUST Cyber Security Framework, URL: https://hitrustalliance.net/ [accessed: 2015-09-15].

[21] HIPAA Administrative Simplification, U.S. Department of Health and Human Services Office for Civil Rights Std., Mar. 2013, URL: http://www.hhs.gov/ocr/privacy/hipaa/administrative/ [accessed: 2015-09-15].

[22] Payment Card Industry Data Security Standard: Requirements and Security Assessment Procedures, PCI Security Standards Council Std., Rev. 3.0, Nov. 2013, URL: https://www.pcisecuritystandards.org/security_standards/ [accessed: 2015-09-15].

[23] Procure Secure: A guide to monitoring of security service levels in cloud contracts, European Union Agency for Network and Information Security (ENISA) Std.

[24] FIPS PUB 46-3 Data Encryption Standard (DES), National Institute of Standards and Technology Std.

[25] FIPS PUB 197 Advanced Encryption Standard (AES), National Institute of Standards and Technology Std.

[26] RDFLib Python library, URL: https://github.com/RDFLib/rdflib [accessed: 2015-09-15].

 ISBN: 978-1-61208-438-1

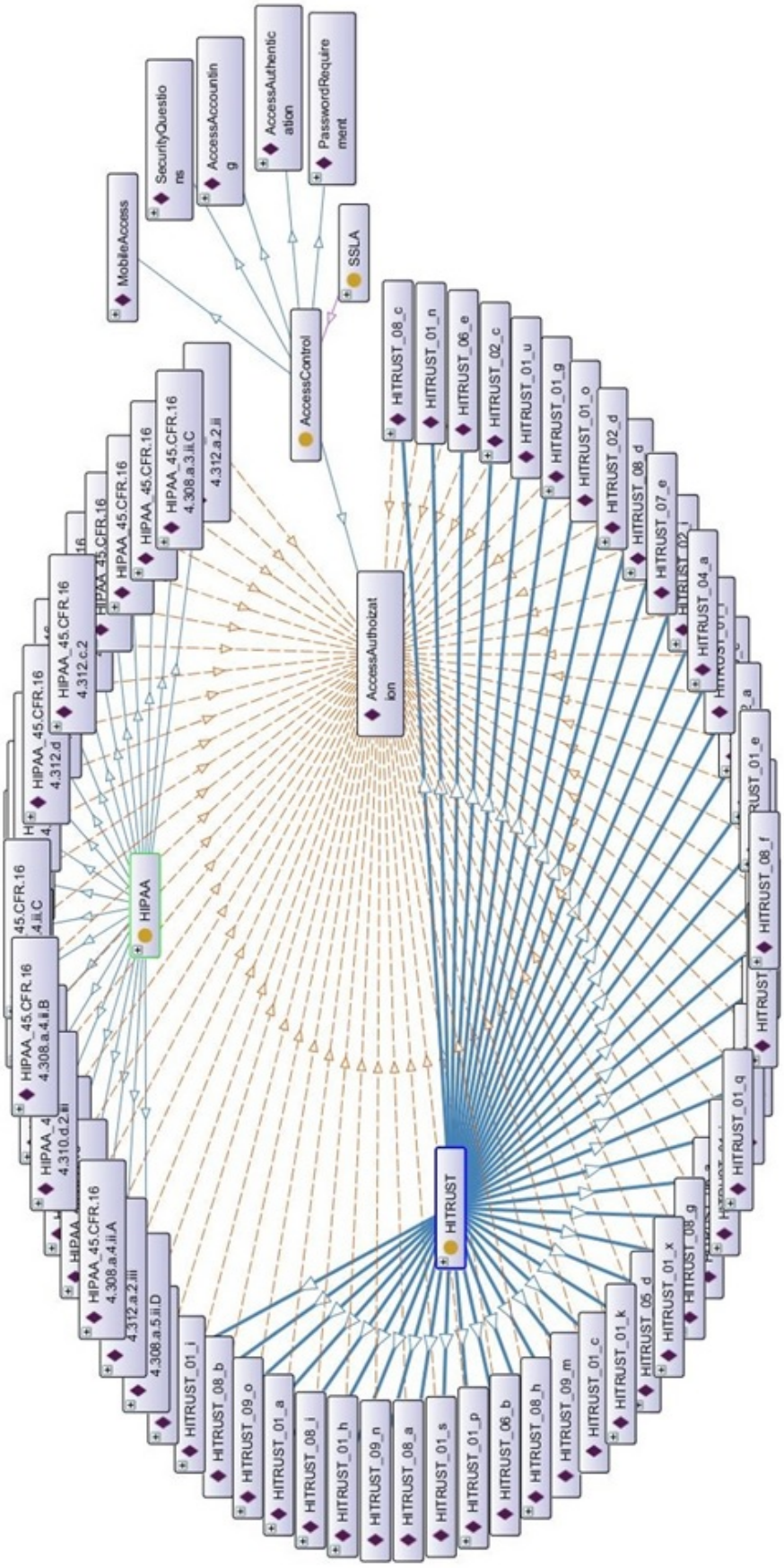


Figure 7. Individuals for both HIPAA and HiTrust in the `AccessAuthorization` class. The left side is for HiTrust and the right side is for HIPAA.

Towards Systematic Safety System Development with a Tool Supported Pattern Language

Jari Rauhamäki, Timo Vepsäläinen and Seppo Kuikka
Department of Automation Science and Engineering
Tampere University of Technology
Finland
Email: {jari.rauhamaki, timo.vepsalainen, seppo.kuikka}@tut.fi

***Abstract*—Design patterns illustrate qualities and features that would suit well in current understanding of safety system development, design and documentation. However, though a number of design patterns for safety system development have been proposed, the focus has been on individual quality attributes such as fault tolerance and reliability. The systematic use of design patterns in the development process has received less attention. In this paper, we discuss and illustrate extended usage possibilities for design patterns as part of safety system development. We discuss a design pattern language that we are developing to cover, e.g., safety system architecture, scope minimization and co-operation with basic control systems. Use of patterns for documentation purposes, tool support for using patterns, and rationale for the pattern approach are discussed as well.**

Keywords-safety system; software; design pattern; safety standard; tool support

I. INTRODUCTION

Design patterns are a means to systematically promote the re-use of design and proven solutions to recurring problems and challenges in design. Each design pattern represents a general, reusable solution to a recurring problem in a given context. Triplets of problems, contexts and solutions are also the essential pieces of information in patterns. In addition, pattern representation conventions can include, among others, relations to other patterns. With such relations describing, for example, rational orders to use patterns, patterns can be combined to collections and to pattern languages. Depending on patterns, the natures of their solution parts can vary too, for example, from source code templates to text and Unified Modeling Language (UML) illustrations.

Software safety functions are software parts of usually multi-technical systems, the purpose of which is to ensure the safety of controlled processes and plants. Unlike many other software systems, safety systems are developed according to standards. The standards govern the development lifecycle activities, as well as techniques and applicable solutions of such systems. However, although design patterns have been specified also for safety system development, their systematic use has not been researched in the domain. This is surprising because the use of patterns could facilitate both design and documentation activities, which are equally important in safety system development.

In this paper, we address the aforementioned issues. The contributions of the paper are as follows. We rationalize how and why design patterns, which have already shown their value in software development, in general [1], could be especially useful in safety system development. We discuss a design pattern language for safety systems, which has been developed and published iteratively and is to be finalized during DPSafe project in collaboration with Forum for Intelligent Machines (FIMA) in the machinery domain. Lastly, we discuss and rationalize the role of tool support in facilitating the use of patterns and in benefitting from patterns.

The rest of this article is organized as follows. Section 2 reviews work related to design patterns and the use of design patterns in safety system development. Section 3 presents a view on the development of software safety systems and rationalizes why and how design patterns could be beneficial. In Section 4, we discuss a design pattern language for safety system development that has been developed at the Tampere University of Technology. Before conclusions, Section 5 discusses the role of tool support when trying to benefit from patterns.

II. RELATED WORK

The design pattern concept was originally presented by Alexander [2][3] in the building architecture domain to refer to recurring design solutions. In software development, design patterns begun to attract interest after the publication of the Gang of Four (GoF) patterns [4]. Thereafter, collections of design patterns have been gathered and used for various purposes in various domains. Results from their use have included, among others, improvements in quality of code, as well as improved communication through shorthand concepts [1].

Design patterns have also been developed for special purposes and application domains, including critical [5] and distributed [6] control systems. In the functional safety domain, especially, patterns already cover many solutions and techniques that are recommended by standards, such as IEC 61508 [7] and ISO 13849 [8]. For example, related to

 ISBN: 978-1-61208-438-1

architecture design in [7], there are patterns to implement redundancy [9] and recovery from faults [10].

Pattern languages, on the other hand, aim to provide holistic support for developing software systems by using and weaving patterns and sequences of patterns [11]. For embedded safety system development, for example, a large collection of (both software and hardware) patterns for various problems is listed in [5]. However, the multi-technical collection is not regarded as a pattern language, per se.

Partially because of reasons to be discussed in the next section, documentation is of special importance in safety system development. A developer of a software safety system needs to be able to prove the compliance of the application to standards. Otherwise, the application cannot be used in the safety system. However, certifiable safety applications are not made by coincidences but by designing the systems and applications systematically, with certifiability in mind. As such, also the software parts need to be specified (modeled) prior to their implementation. On the other hand, the suitable solutions (patterns) that are used in the applications should already be visible in the models. Otherwise, the use of the patterns would not be documented in the models and valuable information could be lost.

It is thus clear that the systematic use of design patterns in safety application development requires tool support for the patterns already in the modeling phase. This is regardless of whether or not the models can be used in producing (automatically) executable code as, e.g., in Model-Driven Development (MDD). Using and applying patterns in UML, which is currently the de-facto software modeling language, has been addressed in several publications. For example, work has been published to specify patterns in a precise manner [12], to apply patterns to models [13, 14], to detect pattern instances [15, 16] and to visualize pattern instances in models and diagrams [17]. However, without extensions the support for patterns is still weak in UML [18].

III. PATTERNS IN SAFETY SYSTEM DEVELOPMENT

The development of safety functions is governed by standards, such as IEC 61508 [7], IEC 62061 [19], and EN ISO 13849-1 [8]. These standards guide the development of safety systems involving electric, electronic and programmable electronic control systems in their operation. Regardless of the variety of standards, we outline a generic development process for safety systems common to the aforementioned standards. The simplified process is illustrated in Figure 1.

The development process begins by the definition of the concepts and scope of the system to be developed. This includes forming an overall picture of the system and defining the boundaries of the system/machine to be analyzed or made safe. The next step is to carry out a hazard analysis and risk assessment. The role of this phase is centric as only known risks can be consciously mitigated. Otherwise risk mitigation measures have no justification. Typically, risk assessment includes hazard identification, risk estimation and evaluation. The former provides an indicator for the risk and the latter assess the impact of the risk, that is, is the risk

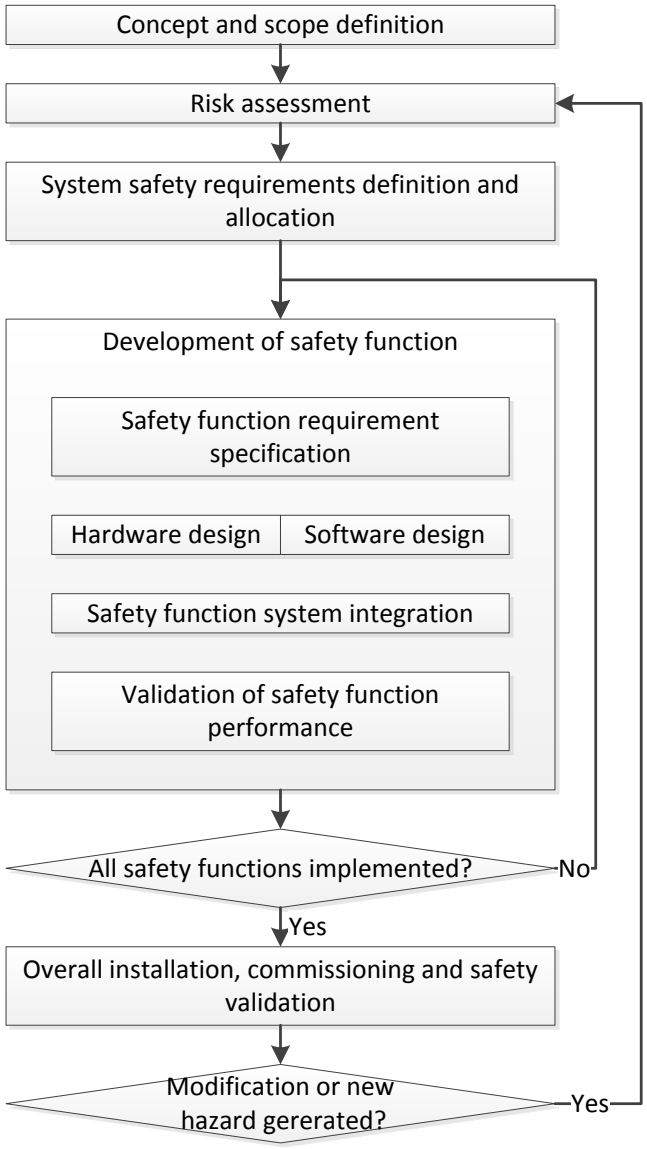


Figure 1. Simplified safety system development process according to EN ISO 13849-1 [8] and IEC 61508 [7]

tolerable or not. Intolerable risks need to be mitigated or made tolerable otherwise.

As the risks are assessed, the requirements considering the system safety can be justifiably made. In this phase, suitable risk reduction methods are selected and their requirements are documented. In the context of this paper it is assumed that the risk reduction method is a protective measure depending on a control system to implement the required functionality. In addition, the allocation of the measures is done. That is, to allocate the measures for dedicated functions.

The next phase is the development (realization in IEC 61508 terminology) of the safety functions allocated in the previous phase. The development process starts with compiling a requirement specification for the safety functions. The specification should include both functional

 ISBN: 978-1-61208-438-1

descriptions, what the functions need to do, and non-functional descriptions, how or within which restrictions the functions need to operate.

Quite often, the non-functional descriptions include the specification of performance or integrity levels for the functions. When the requirement specification is completed, the hardware and software design can begin. In this state the hardware and software parts of the safety function are designed, potentially with separation between the design teams. Thus, hardware and software integration needs to take place along the design process. At this point, a functional entity can be constructed including both the hardware and software to be used in the final system. Finally, the results of the safety function development are verified to match the safety function requirements and required performance/integrity levels. If unimplemented safety functions exist, the development process is reinitialized for the next safety function.

A. *Utilization of patterns in safety system development*

In the context of safety system development and design, design patterns can be used to capture and provide solution models for techniques and applicable solutions that are recommended and/or required by applicable standards. In this case, a design pattern captures the solution that is used in order to fulfill the requirements and recommendations of a standard. Such design patterns can be linked to the parts of the standards for which the design patterns provide a complete or partial fulfillment or help to achieve to fulfill the standard requirements. This kind of approach also supports building the libraries of named solutions. That is, the patterns support the awareness and usage of the solutions.

One can justifiably argue that standard solutions to recurring problems have been applied in safety system development and other domains of engineering for years – without necessarily calling them patterns. However, their unconscious use may not have eased the task of documenting the systems. Since design patterns provide names for solutions, they can be used in communication, too [1]. Though initially applicable to discussions and face-to-face communication, design patterns can be used as a part of written and diagrammatic documentation. This is achieved by referring to the solution illustrated by a pattern with the name of the pattern that should be both illustrative and related to the application context.

The documentation aspect can be achieved by marking the patterns in, e.g., diagrams that are used as a part of the system documentation. This can enhance traceability between the standard solutions and their practical applications in systems. For a pattern-aware person, this may increase the understandability and traceability of the design decisions, too. To take further advantage of this setup, statistics could be gathered to see which patterns are used the most and in which kind of situations. It can also be noted that the quality attributes understandability and traceability are similarly components of systematic integrity acknowledged by IEC 61508 [7].

Other viewpoints supporting the utilization of design patterns in safety system development include for instance [20]:

- Patterns document well-tried solutions and thus condense experience on proven solutions, which is of special importance in the domain. The approach resembles, for instance, the proven in use concept defined by IEC 61508.
- Patterns can alleviate bureaucracy by providing practical solutions and approaches to fulfil requirements given to safety system development in, for example, standards. Bridging the gap between the requirements and design and implementation eases the burden of designers.
- Patterns create the vocabulary of solutions to domains. Assuming that the patterns are known by both the developer and maintainer of a system, patterns can help to communicate the structural and operational principles of the system. This aspect thus improves the communicability and maintainability of the system.

B. *Safety system patterns*

In the context of this paper, we are especially interested in design patterns for safety system development, called safety system patterns here. These patterns are, or at least they are meant to be, most useful in the development of (functional) safety systems. This does not indicate that the patterns could not be used for other purposes as well. However, the contexts of the patterns relate them to the safety system development. It is up to the readers or appliers of the patterns to judge whether the solutions are applicable outside the indented contexts of the patterns, too.

It should be noted that a pattern does not necessarily illustrate the cleverest or the most innovative solution or approach to the defined problem. Instead, the preferable approach is to provide proven solutions and approaches that have been utilized successfully in practice, in real projects and systems. This is, on one hand, targeted to provide assurance on the applicability of the solution, for instance, in the eyes of an inspector. On the other hand, the most innovative solutions might promote other quality attributes than simplicity, which is one of the most important driving qualities behind a safety system development.

So, which parts does a safety system pattern consist of? In our work, we have used a slightly modified canonical pattern format [21]. That is, each pattern documents the context, problem and solution. They are complemented with forces, consequences, example, known usages and related patterns, see Figure 2. The triplet of context, problem and solution provides the main framework for the patterns. These aspects should provide sufficient information to apply a given pattern. However, the other aspects, for instance, support the selection of the most suitable pattern and help to identify other potentially applicable patterns. The former aspect is achieved through the definition of forces and consequences. Forces relate to the context, refine the problem, and direct the solution to the one selected to be illustrated on the pattern. On the other hand, consequences

 ISBN: 978-1-61208-438-1

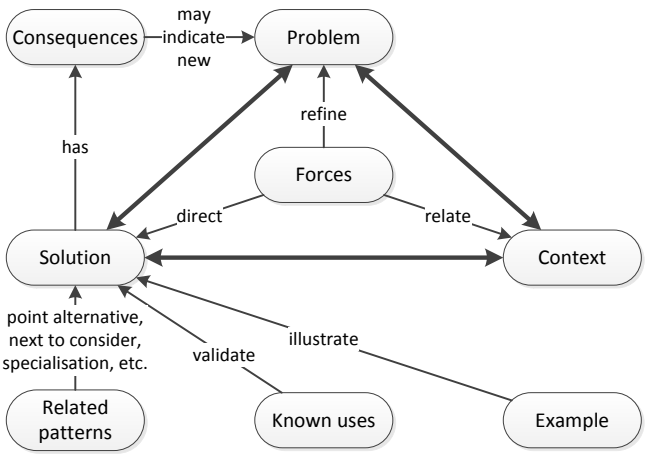


Figure 2. The pattern structure used in our safety system patterns.

provide hints to select a solution proposed by a certain pattern. Presumably one wants to select a pattern or a solution that has the most positive consequences and/or the least negative consequences produced by the solution.

In addition to the mentioned pattern aspects, safety system patterns could be complemented with an aspect indicating the applicable performance level (PL), safety integrity level (SIL), or similar quantity. This is to indicate for which purposes or levels (as defined in standards) the pattern can be used. [21]. For certain patterns or solutions such indicators can be given directly and for others such indicators are indirect or cannot be given at all. For instance, a pattern implementing cyclic execution behavior could be recommended or highly recommended on all safety integrity levels (as defined on IEC 61508-3:2010 table A.2 [7]).

How and where can design patterns then be obtained? Foundationally, design patterns document recurring solutions. The basic assumption is that at least three known usages for a solution need to be obtained to call a solution a design pattern [22]. Keeping this in mind at least the following pattern mining approaches can be considered.

As standards, such as the mentioned IEC 61508 and EN ISO 13849-1, provide requirements considering safety system design and development, they are potential candidates as source information. One potential approach is to take requirement clauses or required techniques or methods and search and provide practical solutions to fulfil the requirements. Depending on the standard and case, the standard may or may not provide instructions on how to actually apply and use required methods, techniques and clauses. Thus treating such elements as problems yields a way to found similar solutions and format them as patterns. For instance, one could consider graceful degradation, which is at least recommended on all SIL levels (as defined by IEC 61508-3:2010 table A.2), and mine patterns to design and implement graceful degradation on software. Using this approach, the integrity (or performance or similar quantity) levels can be directly linked to the patterns.

Literature and similar sources provide a feasible source for pattern mining. Solutions found from different literature sources can be considered pattern input. However, potentially the most credible sources for pattern mining are existing systems and their documentation. In the context of safety system patterns, such sources would be safety systems, their documentation and developers. To provide additional credibility for the mined safety system patterns (at least from the standard point of view), the patterns should be mined from inspected and approved systems. Such merit supports the patterns as the solution has been used as a part of an approved system. It should be noted, however, that a pattern originating from an inspected system does not directly implicate that the new system in which the pattern is applied, would be automatically approved. Nevertheless, such a pattern provides support and trust to believe that the solution is approvable in similar context.

Thus, ideally safety system patterns are mined from existing, inspected, and approved safety systems. As such, the solutions should be applicable on similar integrity level systems and also on lower levels although this is not always the case. Actually, by looking for instance IEC 61508-3 Annex A, this is not always the case. There are methods and techniques highly recommended, e.g., on SIL 3-4 and only recommended on SIL 1-2. Apparently the method or technique is still applicable, but it may be considered too heavy-weight or expensive for the lower integrity levels. To complement this approach, the inspection process and results could be systematically used to document the approved solutions in the form of patterns. During the process, the inspector approves and declines some of the solutions, approaches, and design decisions, which should be considered valuable input for future work. In the end, the inspections cost money and other resources to the customer so it is rational to try to minimize the process and to learn from mistakes and successful designs. Such work would support one of the purposes of patterns in the first place, that is, the systematic reuse of solutions.

IV. A PATTERN LANGUAGE FOR SAFETY SYSTEMS?

First of all, what do we mean by a pattern language? A pattern language is in our case a set of patterns that consider the same domain and are interconnected through relations. According to Eloranta et al., a pattern language is a concept "guiding the designer in building a coherent whole using patterns as building blocks" [6]. In this context, building block mindset, pattern relations and shared domain context between the patterns is seen centric to form the grammar to use the patterns. In practice, the pattern language defines restrictions, rules and suggestions on how to compose the designs of the provided building blocks. [6]. A collection of patterns, in contrast to a pattern language, does not have to have grammar or relations between the patterns.

The relations promote co-usage of the patterns as they guide a designer through the language by providing her with links indicating patterns that can be considered next, alternative, specialized and incompatible solutions related to the pattern that has been recently applied. Although the described approach may ease decision making, it may also narrow the designer viewpoint. A pattern language cannot include all possible solutions and the ones that are included,

 ISBN: 978-1-61208-438-1

do not necessarily introduce the best alternative for a problem or situation under consideration.

One way to utilize the pattern language in design work was described above. The mentioned pattern relation based language walkthrough approach is a rather optimistic view at least if a large context is considered. Safety system development as well as other system development is a process consisting of multiple phases. Covering all of these with a single language of patterns is a large scale problem itself not to mention how to parse a meaningful language by establishing the pattern relations and interconnections. Still, patterns can provide pinpointed solutions to encountered problems and the related patterns may offer ideas during the design process. From our perspective, this is a more feasible use case for a safety system pattern language. To support the usage of the language, the patterns should be, however, grouped so that they resemble the corresponding design phases. That is, architectural patterns would benefit architecture design phase issues and implementation patterns (or idioms) the implementation phase issues.

The safety system design pattern language developed at the Tampere University of Technology has currently some 50 patterns and/or pattern candidates and some of them have been discussed in the workshops of patterns conferences [23]-[27]. (Pattern candidates are initial pattern ideas that do not yet have three known uses, that is, they are under construction. We have found writing pattern candidates an excellent way to communicate the ideas and find new known usages for the pattern candidates.)

In its current state, relations have not been specified for all the patterns of the language, but there are relations between the individual patterns. For example, patterns can specialize more general solutions in stricter contexts. Thus one could say the language lies somewhere between a pattern language and a collection of patterns at the moment. However, our purpose is to develop a full pattern language for safety system development.

We started the work in 2010 and the patterns have been collected, developed and published under various projects such as SULAVA, ReUse, and currently under DPSafe project. In the DPSafe project, we are working with several companies involved one way or another in safety systems design and development in the context of machinery applications. The target of the project is to mine and document design patterns considering software based safety functions and systems as well as gain new known uses for the existing patterns and identified pattern candidates. The participating companies include machinery producers, engineering offices, as well as software houses so there is potential to have different relevant views on the subject.

The patterns are targeted to safety system development. Currently, the language includes patterns and pattern candidates considering, for instance:

- development process
- risk mitigation strategies
- architecture and principles in terms of
  - software
  - hardware
  - system
- co-existence with control system
- scope reduction

In contrast to, for example, redundancy, diversity and other fault tolerance related matters, the sub domains mentioned above seemed to have less attention by pattern community. Thus our purpose is to extend the pattern approach to cover larger part of the safety system development outside the fault tolerance aspect. According to our work carried out in the DPSafe project, there seems to be a clear need for such an approach.

V. On Tool Support for Design Patterns

Whereas some of the benefits of patterns described in Section 3 could be achievable in any case, it is clear that tool support for patterns could increase their benefits significantly. For example, even without tool support, pattern names can become a part of the developer vocabulary [1]. Without a doubt, recurring solutions have also been used in the domain. However, using patterns to improve the traceability of standards solutions, for instance, would certainly benefit from automated functions already during the specification and modeling of the applications. Unfortunately, the support for patterns is in current software modeling tools restricted, at best. The purpose of this section is to discuss opportunities and challenges related to pattern tool support in safety system development. When appropriate, lessons learned from the previous work of the authors [18] will also be provided.

A. On Pattern Modeling

As mentioned, tool support for patterns is currently weak. For example, the pattern concepts of UML, structured collaborations [28], restrict patterns to describe the contents of the UML classifiers only. Thus, elements such as components and packages that would be useful in describing architectural patterns (for instance) cannot be used in patterns in UML [18]. The variety of published patterns in literature, however, covers problems on different levels of design and for various purposes. It cannot be said that all the patterns would be related to classifiers (classes) when all patterns are not even related to software systems. The origin of the (pattern) concept is in building architectures [2, 3] and there are also, for example, multi-technical pattern collections (such as [5]) with both software and hardware aspects. It is thus clear that the UML pattern concepts are currently too restricting, by nature.

With respect to the modeling of multi-technical patterns mentioned above, they could be used in SysML models, which are not restricted to software. However, the use of patterns would not have to be limited to modeling languages at all. For example, patterns could be equally useful in, for example, Computer Aided Design (CAD) tools and software Integrated Development Environments (IDE), in aiding practical design and programming work. Similarly to software engineering, also other engineering disciplines most certainly have recurring problems with known solutions.

While acknowledging this, in our work [18] the focus in developing tool support has been on safety systems and their UML and Systems Modeling Language (SysML) based

 ISBN: 978-1-61208-438-1

modeling in a Model-Driven Development (MDD) context. With new pattern modeling concepts and by integrating them into both UML and SysML, the aim has been to support hardware aspects in addition to software and UML modeling. Safety systems are also systems that are developed and approved as a whole. Good practices and documentation are needed not only for software parts but for all parts of the systems, regardless of their implementation technologies. However, while the developed approach [18] currently allows pattern definitions and instances to consist of practically any modeling elements, the approach suffers from the drawback of not being easily portable to standard tools.

B. On Pattern Instances

In addition to (more or less) formal approaches, e.g., that of UML, modeling tools could support patterns also in an informal manner. Informal support has been developed into, e.g., MagicDraw that enables instantiating patterns from libraries by copying modeling elements. This functionality is not restricted to classifiers as is the case with standard UML. However, copying patterns (informally) can support mainly the aspect of using the solutions and not necessarily using the information about the use of the solutions. Copying model elements may not enable storing information about the elements being part of a pattern instance so that the information could be used for, e.g., documentation purposes.

There is existing research, e.g., [15] and [16], on detecting pattern instances in design models by searching for model structures that are similar to pattern definitions. However, it is questionable whether the use of such work would be an appropriate solution in safety system development. A developer does not use a design pattern by a coincidence. Instead, developers decide to apply patterns because they are facing challenges that they aim to solve with the solutions of the patterns. As such, it is natural that the decisions, which are architectural decisions, should be documented. Why should one try to guess whether a pattern has been applied when the decision could have been explicitly marked in the model when applying the pattern?

Identifying pattern instances based on markings could also be more reliable by nature than trying to detect instances with, for example, the mentioned comparison techniques. When patterns are used in design, they are applied to contexts in which it is feasible to use context specific names and to include additional properties. For example, a non-trivial subject (in an Observer [4] instance) should probably have properties (etc.) that the observer would be interested in. With context specific names, properties and surroundings (in the model), the results of comparisons could be less reliable. However, by marking pattern instances explicitly, the information should be as reliable as documentation is in general. In the end, it would be about the reliability of the developer that marks the pattern instances.

It is thus clear that the information on pattern occurrences should be stored (i.e., the pattern occurrences marked) when they are created. This is also the case in the approach of the authors [18]. Patterns, however, could be in general instantiated both manually and in a tool-assisted manner and the initiatives (to instantiate patterns) could come from either a developer or a tool.

C. On Instantiating Patterns

In a simple, conventional case, pattern instances can be assumed to be always created manually. In this case, it is natural to assume the markings (about the pattern instances) to be created manually, too. Otherwise, a tool would need to – somehow - know about a pattern being applied although the task would be performed by a developer. A tool could also include support for marking the pattern instances - without assisting in the pattern application task itself. However, also in this case the responsibility over the (possibly easily forgotten) marking task should be taken by the developer who knows about the pattern being applied.

Assuming that the pattern application process would be assisted by the tool, also the markings could be on the responsibility of the tool because the tool would know about the application. This thinking has also been used in our work [18]. When patterns are created with an interactive wizard, a developer can justifiably expect the tool to handle the markings. However, markings can be edited (and created) also manually. For example, functions to manually edit markings are needed when deleting or editing a pattern instance.

D. On Initiatives to Instantiate Patterns

In order to *actively* suggest a design pattern to be applied, the tool should have the ability to identify both the context and the problem at hand (in the design task) and to notice that they correspond to the context and problem of the pattern. If the active party was the developer, the tool would not necessarily need to have all the abilities. A set of suggested patterns, to be shown as a response to a user activity for example, could be narrowed down from all possible patterns based on the identification of context or problem. Naturally, with less information, not all the suggestions could be appropriate. However, it would still be up to the developer to make the decision.

Detecting a context of a pattern to match that at hand could be done based on a graph or semantic techniques, for example. However, there could still be challenges in formalizing contexts of many existing patterns that have been defined mainly with text. Identifying a problem, *what the developer would like the system to be like*, could be even more difficult to automate, and prone to errors.

If the active party to initiate an activity to apply a pattern would be the developer, also key words and search functions could be used to filter suggested patterns. This would not be possible if the active party would be the tool, so that the initiative would come prior to any user activity, i.e., prior to typing the key words. In addition, with the key words would come the problem of using different words to describe similar aspects. Nevertheless, key words could provide a sufficiently practical solution for suggesting patterns.

When suggesting patterns to use, a tool could also take advantage on information included - not in the patterns themselves - but in the pattern languages and collections that the patterns appear in. For example, when noticing a pattern

 ISBN: 978-1-61208-438-1

to follow a recently used pattern in a pattern language and the problem of the pattern to match the context at hand, the pattern could be (at least) raised in a list of suggested patterns. Similarly, relations in pattern languages that indicate patterns solving the resulting problems of other patterns could be used in an automated manner to facilitate the work of developers.

In our work [18], pattern suggestions currently based on comparing the patterns that are used in models to collections of patterns that have been formed to correspond to the recommendations of standards. In the domain, this is meaningful since the standards govern and restrict the practical solutions that can (or should) be used by developers. However, the patterns are not yet suggested in any specific phase and the initiative to use patterns comes always from the developer. On the other hand, suggestions do not rely on the identification of either context or problem at hand. This could, however, be a possible future research direction.

In the domain, there can be also competence requirements for developers. As such, it can be assumed that appropriate solutions (patterns) are known by developers and that tool support for suggesting patterns would not even be a necessity. Nonetheless, automated functions can be useful in gathering information on the use of the patterns when there is reliable information about their presence available.

E. On Using Pattern Instances

When pattern instances are reliably detected (marked), the information can be collected from models for analysis purposes or to present it in a tabular, compact form. Especially this can be used to support traceability between solutions and their use, as demonstrated in [18]. Traceability is also a good example property in the (safety) domain because it is a property of systematic integrity and required from safety system development. As discussed in Section 3, the development process of software safety systems and applications consists of phases during which developers should apply appropriate techniques and measures that are to ensure the quality of the applications. Documentation is, though, needed to indicate how and where the techniques and measures have been used.

With pattern marks, it is also possible to automate different kinds of consistency checks, in addition to supporting traceability. For example, it can be made sure that patterns are appropriate for the safety levels required from the safety function or application. Naturally, this requires information on the applicability of the solutions to different levels of safety.

VI. DISCUSSION AND CONCLUSIONS

This paper has discussed the role of design patterns in facilitating the development of software safety systems and applications. Design patterns, which are essentially triplets of contexts, problems and solutions, are a means to systematically re-use design and proven solutions to recurring problems and needs. Their systematic use in the safety system development, however, has not been researched extensively although the re-use of recommended solutions is a general virtue in the domain.

Reasons why design patterns could, in general, benefit safety system development are various. Patterns document proven solutions, which provide designer support on selecting the solution to be used in the safety system under design. Known usages and ideally known usages from inspected and approved systems build this support. Patterns can illustrate practical approaches and solutions to alleviate the requirements considering safety system development given in standards, etc. This eases the burden of the designer by bridging the gap between standards and safety system design and implementation. In relation to this, patterns can be used as a part of documentation.

To provide designers with the patterns to be used in safety system design and development, we have mined and documented a set design patterns and pattern prototypes. The patterns consider various aspects of the safety system design including the development process, architecture, co-existence with basic control systems and scope minimization aspects. The work considering the pattern collection is in progress and current effort is to extend the collection to software based safety functions. New known usages for the existing patterns and pattern candidates are also being collected.

The development of safety systems is a systematic process that is governed by standards. Phases of the process build on information produced in the previous phases so that, for example, safety function requirements are specified to treat previously identified hazards and their associated risks. In the implementation phases of the process, developers are required to apply solutions, techniques and measures that are recommended by the standards and can be assumed to result in sufficient quality. However, in safety system development, it is not enough to apply the required techniques and solutions. Developers need to be able to prove the compliance of the applications to standards. This is where appropriate documentation - including information on the usage of the solutions - is needed.

Clearly, certifiable software parts of safety systems are not built by coincidences but by designing them systematically, with the use of appropriate solutions and techniques. As such, the applications need to be specified prior to their implementation, which usually includes at least their partial modeling. Unfortunately, the support for patterns is in UML, the de-facto software modeling language, restricted at best.

When developing pattern modeling approaches, however, patterns should be specified with dedicated modeling concepts and pattern instances marked in the models. In this way, reliable information on patterns could be used for documentation purposes and to automate consistency checks. In the future, tool support could be developed also for assisting developers in selecting patterns to use. However, this task should perhaps consider not only information included in the patterns themselves but also the information included in pattern languages and collections of patterns. Such collections could then be developed with the requirements of safety standards in mind.

 ISBN: 978-1-61208-438-1

REFERENCES

[1] K. Beck, et al., "Industrial experience with design patterns," in Proceedings of the 18th International Conference on Software Engineering, 1996, pp. 103-114.

[2] C. Alexander, S. Ishikawa, and M. Silverstein, Pattern languages. Center for Environmental Structure, vol. 2, 1977.

[3] C. Alexander, The timeless way of building. Oxford University Press, 1979.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

[5] A. Armoush, Design Patterns for Safety-Critical Embedded Systems. Ph.D. thesis, Aachen University, 2010. Available http://darwin.bth.rwth-aachen.de/opus3/volltexte/2010/3273/pdf/3273.pdf [referenced 25.6.2015].

[6] V. Eloranta, J. Koskinen, M. Leppänen, and V. Reijonen, Designing Distributed Control Systems: A Pattern Language Approach. Wiley Publishing, 2014.

[7] IEC, 61508: functional safety of electrical/electronic/programmable electronic safety-related systems. International Electrotechnical Commission, 2010.

[8] ISO, 13849-1:2006 Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design. International Organization for Standardization, 2006.

[9] B. P. Douglass, Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems. Addison-Wesley, 2003.

[10] R. Hanmer, Patterns for Fault Tolerant Software. John Wiley & Sons, 2013.

[11] F. Buschmann, K. Henney, and D. Schimdt, Pattern-Oriented Software Architecture: On Patterns and Pattern Language. John Wiley & Sons, 2007.

[12] R. B. France, D. Kim, S. Ghosh, and E. Song, "A UML-based pattern specification technique", Software Engineering, IEEE Transactions On, vol. 30, 2004, pp. 193-206.

[13] P. Kajsa and L. Majtás, "Design patterns instantiation based on semantics and model transformations", in SOFSEM 2010: Theory and Practice of Computer Science, Springer, 2010, pp. 540-551.

[14] R. France, S. Chosh, E. Song and, D. Kim, "A metamodeling approach to pattern-based model refactoring," IEEE Software, vol. 20, 2003, pp. 52-58.

[15] A. Pande, M. Gupta, and A. K. Tripathi, "A new approach for detecting design patterns by graph decomposition and graph isomorphism," in Contemporary Computing, Springer, 2010, pp. 108-119.

[16] N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidis, "Design pattern detection using similarity scoring," Software Engineering, IEEE Transactions on, vol. 32, 2006, pp. 896-909.

[17] D. Jing, Y. Sheng, and Z. Kang, "Visualizing design patterns in their applications and compositions", Software Engineering, IEEE Transactions on, vol. 33, 2007, pp. 433-453.

[18] T. Vepsäläinen and S. Kuikka, "Safety patterns in model-driven development," The 9th International Conference on Software Engineering Advances (ICSEA 2014), Nice, France, 2014, pp. 233-239. ISBN: 978-1-61208-367-4.

[19] IEC, 62061: Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems. International Electrotechnical Commission, 2005.

[20] J. Rauhamäki, T. Vepsäläinen, and S. Kuikka, "Patterns in safety system development", The Third International Conference on Performance, Safety and Robustness in Complex Systems and Applications (PESARO 2013), 2013, pp. 9-15.

[21] B. Appleton, "Patterns and software: Essential concepts and terminology", Object Magazine Online, vol. 3, no. 5, 1997, pp. 20-25.

[22] C. Kohls and S. Panke, "Is that true...?: thoughts on the epistemology of patterns". In Proceedings of the 16th Conference on Pattern Languages of Programs (PLoP '09). ACM, New York, NY, USA, Article 9, 2009, 14 pages. http://doi.acm.org/10.1145/1943226.1943237.

[23] J. Rauhamäki and S. Kuikka, Strategies for hazard management process. The 19th European Conference on Pattern Languages of Programs (EuroPLoP 2014), 9.-13.7.2014, Irsee, Germany, ACM New York, NY, USA 2014. Article 31. DOI: 10.1145/2721956.2721966. ISBN: 978-1-4503-3416-7.

[24] J. Rauhamäki and S. Kuikka, Patterns for Sharing Safety System Operation Responsibilities between Humans and Machines. The VikingPLoP 2014 Conference, 10.-13.4.2014, Vihula, Estonia, 2014. ACM New York, NY, USA, 2014, pp. 68-74.

[25] J. Rauhamäki and S. Kuikka, Patterns for control system safety. The 18th European Conference on Pattern Languages of Program, EuroPLoP 2013, Irsee, Germany, July 10-14, 2013. ACM, 2013, Article 23. DOI: 10.1145/2739011.2739034, ISBN 978-1-4503-3465-5.

[26] J. Rauhamäki, T. Vepsäläinen, and S. Kuikka, Patterns for safety and control system cooperation. In: Eloranta, V.-P., Koskinen, J. & Leppänen, M. (eds.). Proceedings of VikingPLoP 2013 Conference, Ikaalinen, Finland 21.3. - 24.3.2013.Tampere University of Technology. Department of Pervasive Computing. Report 2, 2013, pp. 96-108.

[27] J. Rauhamäki, T. Vepsäläinen, and S. Kuikka, Functional safety system patterns. In: Eloranta V.-P., Koskinen, J., Leppänen M. (eds.). Proceedings of VikingPloP 2012 Conference, 17.-20.3.2012. Tampere University of Technology. Department of Software Systems. Report. Nordic Conference of Pattern Languages of Programs vol. 22, Tampere, Tampere University of Technology. 2012, pp. 48-68. Available: http://URN.fi/URN:ISBN:978-952-15-2944-3.

[28] OMG, Unified Modeling Language Specification 2.4.1: SuperStructure. Object Management Group, 2011.

 ISBN: 978-1-61208-438-1

An Analysis of Seven Concepts and Design Flaws in Identity Management Systems

João José Calixto
Cesar.edu
CESAR – Center for Advanced Studies and Systems
Recife, Brazil
e-mail:calixtounicap@gmail.com

Felipe Silva Ferraz
Informatics Center
Federal University of Pernambuco
Recife, Brazil
e-mail:fsf3@cin.ufpe.br

***Abstract* –Identity management uses models to accredit, manage and use digital identities. These models connect isolated islands of authentication and authorization systems in a federated system. However flaws in the design and concept of these models, such as identity theft and even users' lack of confidence in truly using these models, can lead systems that use its benefits to being non-successful on the market. This article presents an analysis of seven design and concept flaws of the identity management model of the main tools on the market, including Security Assertion Markup Language (SAML), OpenID, Microsoft CardSpace and an academic framework called Inter-Cloud Identity Management (ICEMAN).**

Keywords-Identity Management; flaws; Identity; design; Security.

I. INTRODUCTION

The huge transformation that cloud computing prompted within the IT industry made software development as a service more attractive [1]. This large-scale paradigm cut out the need for large investments [2]. The transparency of the services provided by the cloud is a key point of this supply-side paradigm [3].

Cloud computing combines virtualization and service-oriented architecture (SOA) in order to provide shared services with regard to computing, data storage, software, applications or for a business [4][5]. However, the resource capacity of a single cloud is finite, so cloud computing has been migrating to a perspective of InterClouds, namely an environment in which several clouds can be configured that can communicate with each other and share data and services.

There are identification mechanisms for each service hosted in cloud computing environments and these make use of solutions for user authentication. However, this approach leads to user fatigue as users must memorize logins and passwords [6]. A study in 2007 on password habits showed that typical web users have on average 27 accounts that require a password, and they type eight passwords per day [7]. Therefore this results in users registering similar or even identical logins and passwords for different types of services [7]–[9]. Another problem associated with user authentication and identification is the disclosure of users' personal information after they are successfully identified in a service.

In this scenario, the identity management (IdM) is needed to mitigate and resolve some of these issues. IdM is a set of technologies and processes that enable computer systems to distribute identity information and delegate tasks by using one or more domains with more security [4][10]. Identity management in cloud computing environments is primarily responsible for authenticating users and supporting access based on his/her attributes. IdM for InterClouds can be represented by a single authentication system can be deployed in heterogeneous clouds [11].

Identity management systems are complex and offer all parties involved, powerful features so as to facilitate the mechanism for identities, credentials, personal information, and to present such information to third parties. These systems can bring about potential failures [12].

This article studies major flaws in the concept, usability and design of the most popularly successful identity management systems on the market, namely OpenID [13], Security Assertion Markup Language (SAML) [14], Microsoft CardSpace InfoCards [15] and an academic framework called Inter-Cloud Identity Management (ICEMAN) [16].

The paper is organized as follows. Section 2 gives a short overview of identity management. Section 3 describes the seven flaws of design in identity management systems, while Section 4 discusses the identity models themselves and their flaws are the topic of Section 5. Finally, in Section 6, conclusions are drawn and recommendations outlined.

II. IDENTITY MANAGEMENT

An identity is defined by an entity or group of entities (a person, computer, organization, etc.) represented solely within a specific scope. Yet much can be derived from the definition. Which are entities and how each identity be uniquely identified? Entities may be objects, or, as in most cases a personal identity.

In each context we have different attributes that make up the identity of how we ourselves are identified. What identifies us are the attributes we possess. Different attributes of identity lead to different entities being identified. In such contexts, we can assume an identity, such as a driver's license number coupled with an the 2-letter code of a Brazilian state. Another simple example is our national, Brazilian ID, which has a numeric record and a

 ISBN: 978-1-61208-438-1

fingerprint. All these identities cited are merely a set of attributes that if not inserted in a context and certified lose their objective, which is assertively to identify the user who gives such information is who that user purports to be. In this scenario, we can perform an analogy with our digital identities, which consist of identifying attributes such as login and password, which, if not inserted in the correct context, are not valid.

Identity management, or IdM, consists of the process and all technologies associated with this to accredit, manage, and use digital identities [17]. In the most common models for an identity management, three parties are highlighted: users, identity providers (IdP) and relying Parties (RP) [4][18].

There are centralized identity models, ie where there is only one authority as IdP that performs authentication and authorization actions and there are also decentralized models, which have more than one IdP [19]. Some examples of decentralized identity management systems are the OpenID, SAML and Microsoft CardSpace. In this article, we will focus on non-centralized identity management systems because they do not require a previous relationship between RP and IdP.

III. SEVEN FLAWS OF CONCEPT AND DESIGN

To be successful in the market, identity management systems must win the trust of users and RPs. For this to occur, the systems must improve security, simplify the control of the flow of personal information, and most important of all, simplify process for authenticating, identifying and checking credentials. The seven failures presented below are topics that should be addressed so that the public absorbs the use of identity management systems to a greater extent [12].

A. Identity management is not the main goal

A user simply wants to utilize the functionality of his/her website. Identity management systems should aim to facilitate those tasks by including features such as security and privacy, but these features that are aggregated with an IdP are considered secondary. Usually functions that offer long-term gain are less valued [20]. Some identity management systems offer time saving features, such as automated form-filling, simplification such as single sign-on, or high-value reputation, all of which can be leveraged across many sites. However, these benefits are often perceived as "secondary" [20].

B. Users follow the path of least resistance

The key to maximizing the direct cost is to construct systems that are easily adopted. This includes processes of authentication and interface with the password, which should become easier compared to current standards. When the technology interferes with desired activities, users tend to create shortcuts to circumvent the security embedded in the process [21][22]. For the success of identity management systems to be successful, users should find them easy, accurate and safe to use them and configure them.

C. Cognitive Scalability is as equally important as technical scalability

Today users undergo so-called password fatigue. They have approximately 25 accounts and they can type 8 passwords a day [7]. To avoid burdening their memory in this way, users generally choose the same logins and similar passwords for various accounts they use [23]. Focusing on cognitive scalability is one of the keys to success. Designing the application only by thinking of one IdP should be avoided. Instead, the designer should analyze the system as a whole.

D. The user's consent may lead to maximum disclosure of information

Many identity management schemes describe themselves user-centric, whereby users or customers have to give their consent so that certain transactions may occur [24]. However, surveys show that when warning messages are displayed consecutively to users, they only read them only superficially and move quickly on so as to achieve their goals, thus jeopardizing their privacy and possibly disclosing unnecessary information to third parties [25]–[27].

An identity management system should provide the uses with more control of the data that they are disclosing, without overloading them and even less without doing so in an uncontrolled way.

E. There is a need for mutual authentication (not just user authentication)

Many identity management models focus mainly on authenticating the user [12]. These types of models can be susceptible to phishing attacks [22]. With software support, attackers can easily simulate the interface of a web site, put in sections that require authentication and steal the user's credentials [28].

In this scenario, what is needed is to authenticate both the RP and the IdP, thereby performing a mutual authentication. This indicates that possibly the conduct of spoofing and phishing attacks can be hampered.

F. RPs want to control the user's experience

In general, for the purposes of monitoring or tracking of users' activities RPs tend to want to control the actions that users perform. However, when an identity management system is used, these steps can be lost and there can be a marked difference between the RP layout and that of the IdP.

To make this transition smooth, it is possible to use the IdP before entering the RP layout, thus hiding that there is communication between the RP and the Idp. The Verisign's OpenID Seatbelt use this strategy.

 ISBN: 978-1-61208-438-1

G. *Trust must be earned*

The decision on to whom users may entrust sensitive data is an extremely difficult one. Various models lead to different authentication requirements and assignments of responsibility. Even the IdP of large corporations may contain vulnerabilities or may be poorly implemented. There are differing privacy policies and business models. No organization can guarantee a completely secure system. Systems designers should have their applications evaluated by specialized security companies before launching systems on the market.

IV. IDENTITY MANAGEMENT SYSTEMS

This section gives a general description of the main flow of authenticating the identity management systems examined in the article.

A. *SAML*

SAML is an XML-based framework for representing and exchanging of security information [29]. The use of SAML for an identity management system follows a flow that differs from the current identification process based on login and password. An RP that groups several services wants the user of each service offered to be identified and authorized. Therefore, the RP must have an IdP and from that moment on, all users must register and identify themselves to that IdP.

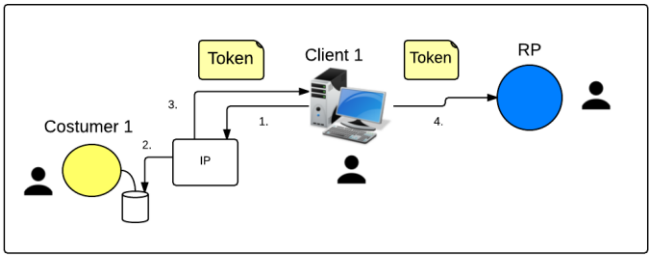


Figure 1. Authentication-flow with SAML.

The IdP will consult a database containing information about the user and will return a SAML token that represents the user identified. This token have the user's attributes such as his/her age, gender and name [30]. Figure 1 illustrates the authentication flow with SAML.

B. *OpenId*

Also based on the Single Sign On (SSO) is the OpenId identity management model [31]. In this model the RP must rely on information from the OpenId provider (OP), the IdPs of the OpenId. Each identifier is represented by a URL, which is unique to each OP so as to reduce collisions between identical URLs [32]. The base authentication flow in the OpenId has the following steps:

1. The user wants to login with RP and inserts his/her OpenID identifier.
2. Using information contained in the handle the RP discovers the OP of the Original.
3. RP connects to the OP using a secret shared between the two parties.
4. RP redirects the user to the OP, which checks its information and redirects to the RP.
5. The user cross-checks information shared with the OP in step 3 with data that the user obtains after step 4.

Figure 2 illustrates the base authentication flow of OpenID.

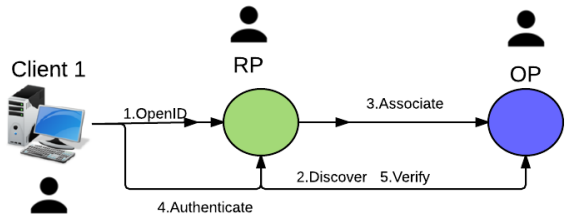


Figure 2. Base authentication flow of OpenId.

C. *Microsoft Card Space*

Microsoft CardSpace (formerly known as InfoCard) was built to give users a conscious digital identity [33]. Since CardSpace is an XML-based framework, CardSpace plug-ins for browsers other than Microsoft Internet Explorer can also be developed, such as the Firefox Plug-in [34]. The framework is based on the identification process users experience in the real world when using physical identification cards CardSpace uses collections of cards, presented in software, which has a similar design to that of a portfolio called identity selector [5]. Each card represents an identity. When an SP searches for an identity the user chooses which card he/she will use from the identity selector [35]. When the SP requires an attribute of the identity, a set of data corresponding to the user's choice is sent to the SP [33].

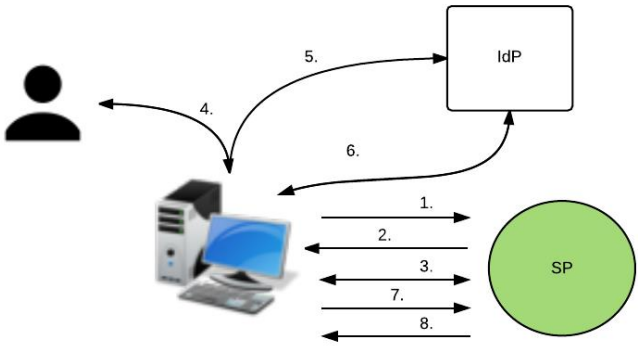


Figure 3. CardSpace Flow.

Figure 3 provides a simplified sketch of the CardSpace framework. In step 1, de the CardSpace enables the user agent or the Service Requestor. In step 2, using a public key the RP identifies itself. After recognizing that the RP is CardSpace- enabled, the CardSpace Enable User Agent (CEUA) retrieves the RP security policy in step 3. In step 4,

 ISBN: 978-1-61208-438-1

the CEUA matches the RP's security policy with the InfoCards that the user has. The user performs an authentication process with the IdP in step 5. If the authentication process succeeds, step 6 takes place, in which the CEUA asks the IdP to provide a security token that holds an assertion of the truth of the claims listed within the selected InfoCard. Finally, the CEUA forwards the security token to the RP in step 7, and, if the RP verifies it successfully, the service will be granted in step 8 [34].

D. ICEMAN

ICEMAN differs from the traditional approach, which has only one IdP for an SP or RP, which is an unreal environment in interclouds. This academic framework proposes a more suitable scheme for interclouds. ICEMAN provides a high interoperability mechanism between any pattern of identity thus facilitating the management of the life flow of the authentication [12]. However, this architecture is still being developed, thus preventing further analysis of the seven failures. Nevertheless, the ICEMAN model for identity management was included in the article as it has a mechanism that can come to add more than one identification and authorization model. Such an approach may ultimately unite existing models, which may be able to mitigate weaknesses and strengthen strong points [16].

V. IDENTIFYING FAULTS IN IDENTITY MANAGEMENT MODELS

We have chosen four Identity Management Systems for our analysis and seven de design flaws which either have dominant positions in Identity Management scenarios or introduced a novel concept which is worth exploring.

1. **Identity management is not the main goal:**

The MS CardSpace model was considered to have the first flaw since it adds a new software to the user's standard way to access information and services. Microsoft has discontinued their CardSpace project. However, we have opted to include it into our analysis because of its fundamentally novel concept of how Identity is presented.

2. **Users follow the path of least resistance:**

It was considered that all models display some difficulty when it comes to installing and configuring them for use. The very concept of the SAMU follows an alternative flow that does not allow the user to follow the path of least resistance.

3. **Cognitive Scalability is as equally important as technical scalability:**

Cognitive scalability in all but the ICEMAN is adequate. The ICEMAN is a framework for better integration of identity management in InterClouds. The scalability of technical cognition scalability does not follow the average of the other models presented.

4. **The user's consent may lead to maximum disclosure of information:**

On the consent of the information to be passed to the user MS CardSpace user is well ahead. However, it is important to emphasize that the type of approach to maintain management of cards can be stressful for users and can generate a new kind of dissatisfaction with the tool. In the case of SAML, in the basic flow of authentication there is nowhere that will say what information can be accessed by the service.

5. **There is a need for mutual authentication (not just user authentication):**

There is the possibility of phishing and spoofing in the identity models. Therefore, it was considered that all configuration management models contain such flaws, which leads the parties involved to add other security mechanisms to mitigate these vulnerabilities [35].

6. **RPs want to control the user's experience:**

No model analyzed initially presents monitoring of the user's actions on the site and the transition between the layout of authentication between IdP and the Client is not specified in any model. Thus, it was assumed that all flows present this flaw.

7. **Trust must be earned:**

On models with greater maturity and interaction with the market, it has been identified that users place greater trust in these. It was considered that the ICEMAN has such a flaw. However, according to research carried out on regular Internet users, it was shown that there is still no confidence in service providers that use MS CardSpace [34].

Table 1 illustrates the results of a comparison between flaws and models.

TABLE I. RESULTS OF A COMPARISON BETWEEN MODELS AND FLAWS.

Flaw	MS Card	OpenId	SAML	ICEMAN
1	✓	X	X	partial
2	partial	✓	✓	✓
3	X	X	X	✓
4	partial	✓	✓	✓
5	✓	✓	✓	✓
6	✓	✓	✓	✓
7	✓	X	X	✓

VI. CONCLUSION

Identity management systems are not just systems for authenticating and authorizing identities but are also a set of methods and procedures that can contribute to greater user immersion within a system which uses, for example, the single sign on. However, some identity management systems failed at least partly because they ignored the topics discussed in this paper.

An overview was given of the most popular identity management systems in the market, namely: OpenId, MS CardSpace, SAML and an academic framework called ICEMAN. Seven flaws in the concept and design of identity management in these systems were analyzed. The flaws

 ISBN: 978-1-61208-438-1

found in the models were compared in a critical analysis of the study of their concept study and how the user can achieve greater reliance on the technology, and the identity management process.

In this article, problems to do with the lack of control in the process of identifying and authorizing users were listed, in addition to flaws in the concept of identity management systems. The study found that the lack of commitment to dealing with the flaws can result in large projects being poorly received by the current market. Strategies to mitigate and solve the problems discussed in the article were also discussed.

Finally, we intend to examine flaws in identity management in greater depth in future studies, which will focus on aspects of privacy, availability and integrity. We would also like to add new systems to the market and to put forward new academic frameworks.

REFERENCES

[1] A. Acquisti and J. Grossklags, “Privacy and rationality in individual decision making,” IEEE Secur. Priv. Mag., vol. 3, no. 1, pp. 26–33, Jan. 2005.

[2] G. Pallis, “Cloud computing: The new frontier of internet computing,” IEEE Internet Computing, vol. 14, no. 5. pp. 70–73, 2010.

[3] A. Gopalakrishnan, “Cloud Computing Identity Management Online security concerns are on the rise and what cloud needs now,” vol. 7, no. 7, pp. 45–55, 2009.

[4] D. Núñez, I. Agudo, P. Drogkaris, and S. Gritzalis, “Identity Management Challenges for Intercloud,” pp. 198–204.

[5] E. Maler and D. Reed, “The venn of identity: Options and issues in federated identity management,” IEEE Secur. Priv., vol. 6, no. 2, pp. 16–23, 2008.

[6] “Password fatigue - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Password_fatigue. [Accessed: 12-Oct-2015].

[7] F. , N. , and H. Shannon. "Technology Corner: Brute Force Password Generation--Basic Iterative and Recursive Algorithms." Journal of Digital Forensics, Security and Law 6.3 (2011): 79-86.

[8] S. Gaw and E. W. Felten, “Password management strategies for online accounts,” in Proceedings of the second symposium on Usable privacy and security - SOUPS ’06, 2006, p. 44.

[9] R. Chow, Ori Eisen, et al. "The future of authentication." IEEE Security & Privacy 1 (2012): 22-27.

[10] E. Maler and D. Reed. "The venn of identity: Options and issues in federated identity management." IEEE Security & Privacy 2 (2008): 16-23.

[11] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, “Security and Cloud Computing: InterCloud Identity Management Infrastructure,” 2010 19th IEEE Int. Work. Enabling Technol. Infrastructures Collab. Enterp., pp. 263–265, 2010.

[12] R. Dhamija and L. Dusseault, “The seven flaws of identity management: Usability and security challenges,” IEEE Secur. Priv., vol. 6, no. 2, pp. 24–29, 2008.

[13] “Final: OpenID Authentication 2.0 - Final.” [Online]. Available: http://openid.net/specs/openid-authentication-2_0.html. [Accessed: 12-Oct-2015].

[14] “XACML SAML Profile Version 2.0.” [Online]. Available: http://docs.oasis-open.org/xacml/xacml-saml-profile/v2.0/xacml-saml-profile-v2.0.html. [Accessed: 06-Nov-2015].

[15] K. Cameron and J. Michael. "Design rationale behind the identity metasystem architecture." ISSE/SECURE 2007 Securing Electronic Business Processes. Vieweg, 2007. 117-129.

[16] G Dreo, M Golling, et al. "ICEMAN: An architecture for secure federated inter-cloud identity management." Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, 2013.

[17] G. Alpár and J. H. Johanneke, “The Identity Crisis Security , Privacy and Usability Issues in Identity Management,” pp. 1–15, 2011.

[18] D. W. Chadwick, “Federated Identity Management,” vol. 5705, pp. 96–120, 2009.

[19] S. Dongwan , A. Gail-Joon and S. Prasad, “Ensuring information assurance in federated identity management,” in IEEE International Conference on Performance, Computing, and Communications, 2004, 2004, pp. 821–826.

[20] A. Acquisti and J. Grossklags, “Privacy and rationality in individual decision making,” IEEE

 ISBN: 978-1-61208-438-1

Secur. Priv. Mag., vol. 3, no. 1, pp. 26–33, Jan. 2005.

[21] A. Adams and M. A. Sasse, “Users are not the enemy,” Commun. ACM, vol. 42, no. 12, pp. 40–46, Dec. 1999.

[22] U. C. Berkeley, “Why Phishing Works,” 2006.

[23] B. M. Gross and E. F. Churchill, “Addressing Constraints: Multiple Usernames, Task Spillage and Notions of Identity,” in CHI ’07 extended abstracts on Human factors in computing systems, 2007, pp. 2393–2398.

[24] A. Cavoukian, “7 Laws of Identity - The Case for Privacy-Embedded Laws of Identity in the Digital Age,” Technology, no. 30 January 2008, p. 24, 2006.

[25] N. Good, R. Dhamija, J. Grossklags, D. Thaw, S. Aronowitz, D. Mulligan, J. Konstan, and S. Hall, “Stopping Spyware at the Gate : A User Study of Privacy , Notice and Spyware Definition of Spyware,” pp. 1–10.

[26] J. Grossklags and N. S. Good, “Empirical Studies on Software Notices to Inform Policy Makers and Usability Designers,” in Financial Cryptography and Data Security, 2008, pp. 341–355.

[27] D. A. Norman, “Design rules based on analyses of human error,” Commun. ACM, vol. 26, no. 4, pp. 254–258, Apr. 1983.

[28] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer, “The Emperor’s New Security Indicators,” 2007 IEEE Symp. Secur. Priv. (SP ’07), 2007.

[29] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, “Formal Analysis of SAML 2.0 Web Browser Single Sign-on: Breaking the SAML-based Single Sign-on for Google Apps,” Proc. 6th ACM Work. Form. Methods Secur. Eng., pp. 1–10, 2008.

[30] P. Arias Cabarcos, F. Almenarez Mendoza, A. Marin-Lopez, and D. Diaz-Sanchez, “Enabling SAML for Dynamic Identity Federation Management,” Wirel. Mob. Networking, Proc., vol. 308, pp. 173–184, 2009.

[31] “Pros and Cons of OpenID - O’Reilly Radar.” [Online]. Available: http://radar.oreilly.com/2007/02/pros-and-cons-of-openid.html. [Accessed: 08-Nov-2014].

[32] “What is OpenID?.” [Online]. Availablehttps://openid.net/get-an-openid/what-is-openid. [Accessed: 05-Nov-2015].

[33] S. Gajek, J. Schwenk, M. Steiner, and C. Xuan, “Risks of the cardspace protocol,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, vol. 5735 LNCS, pp. 278–293.

[34] W. A. Alrodhan and C. J. Mitchell, “Addressing privacy issues in CardSpace,” in Third International Symposium on Information Assurance and Security, 2007, pp. 285–291.

[35] V. Bertocci et al.,Understanding Windows CardSpace An Introduction to the Concepts and Challenges of Digital Identities Technical Reviewers.

 ISBN: 978-1-61208-438-1

ATM Security

A Case Study of a Logical Risk Assessment

Johannes Braeuer
Dept. of Information Systems
Johannes Kepler University
Linz, Austria
email: johannes.braeuer@jku.at

Bernadette Gmeiner
Banking Automation
KEBA AG
Linz, Austria
email: gmb@keba.com

Johannes Sametinger
Dept. of Information Systems
Johannes Kepler University
Linz, Austria
email: johannes.sametinger@jku.at

***Abstract*—Automated Teller Machines (ATMs) contain considerable amounts of cash and process sensitive customer data to perform cash transactions and banking operations. In the past, criminals mainly focused on physical attacks to gain access to cash inside an ATM's safe. They captured customer data on the magnetic strip of an ATM card with skimming devices during insertion of the card. These days, criminals increasingly use logical attacks to manipulate an ATM's software in order to withdraw cash or to capture customer data. To understand the risks that arise from such logical attacks, we have conducted a risk assessment of an ATM platform that is running in a real banking environment. The result of this assessment has revealed the main issues that are responsible for vulnerabilities of an ATM platform. In this paper, we discuss the findings of our risk assessment as well as countermeasures to mitigate serious risks in order to ensure a secure banking environment. The risk assessment has revealed effective countermeasures and has additionally provided a prioritization of activities for ATM manufacturers.**

Keywords-automated teller machines; ATM security; embedded systems; risk assessment.

I. INTRODUCTION

Automated Teller Machines (ATMs) have their roots back in the late 1930s, but they began to revolutionize the banking environment in the 1960 [1]. With the integration of real-time terminals, ATMs have been developed to data processing units that contained commercially available computers. Today, almost all three million ATMs around the world are running on operating system (OS) Windows [2]. On top of Windows, the ATM platform controls all peripheral devices and uses the OS to communicate with device drivers. The ATM platform also provides an interface to multi-vendor ATM software, i.e., bank applications that utilize the platform's functionality. Besides Windows, ATMs use the Internet Protocol (IP) for communication in the banking network [3]. Consequently, the ATM network is part of the banking network, which in turn is part of the Internet. ATMs have developed from stand-alone equipment with simple cash dispensing capabilities to a network of connected devices for bank transactions. ATMs contain a remarkable amount of cash for their daily operation. Thus, they have always been an attractive target for thieves and fraudsters [4]. Also, they were available around the clock and often located off-premises [5]. Fraudulent activities are not only attracted by cash, but also by data that is required to conduct full bank transactions. Risk assessments provide information to select adequate countermeasures and controls for mitigating the likelihood or impact of risks. We have conducted such a risk assessment concentrating on logical risks of an existing ATM platform. The proposed method can easily be extended to physical risks and risks resulting from card and currency fraud.

In this paper, we will first provide an overview of attacks to ATMs as well as their countermeasures. We will then evaluate the countermeasures for logical attacks by a risk assessment. As a result, we can confirm that suggested countermeasures work for the identified risks. Additionally, we can prioritize these countermeasures and provide a guideline for those responsible for ATM security. The paper is structured as follows. In Section II, we describe criminal activities in the context of ATMs and discuss traditional attacks and countermeasures. Section III concentrates on logical ATM security. Section IV presents a risk assessment approach, which is then used in Section V to determine the risks of an ATM platform. Findings are discussed in Section VI. Related work and a conclusion follow in Sections VII and VIII, respectively.

II. AUTOMATED TELLER MACHINES

An ATM is a cash dispensing machine with the capability to credit or debit a customer account without human intervention [1]. The term ATM has been used synonymously for cash machines, cash dispensers or cash recyclers. However, the designation ATM is inappropriate when a machine cannot perform a complete financial transaction initiated by the customer. Thus, ATMs support synchronous or asynchronous electronic data processing operations in an online and real-time manner [1]. ATMs have revolutionized the banking sector. Their widespread dissemination has grown to a world-wide use of around 2.8 million ATMs. This number is expected to reach 3.7 million by 2018 [6]. ATMs have always been an attractive target for thieves [4]. Reinforced by the fact that ATMs are typically available 24/7 and often located off-premises, they are vulnerable to cash thefts [5]. However, ATM crime, including ATM fraud, goes beyond stealing cash. Illegally obtaining customer's personal information, such as bank account data, card number and PIN is an additional security issue that is related to ATMs [5][7]. These digital assets do not provide an immediate profit, but they can be sold on illegal credit card data markets on the Internet [8].

 ISBN: 978-1-61208-438-1

There are three different types of attacks, i.e., card and currency fraud, physical attacks and logical attacks [9][10]. Various Information Technology (IT) security standards have been developed and vendors have recommended security concepts pertaining to ATMs [11]. The goal is to secure an entire ATM and its environment. Similar to ATM crime, ATM security can be divided into the three different core areas card and currency protection, physical security, and logical security. The former two will be described in the next subsections. Logical ATM security will follow in Section III.

A. Card and Currency Fraud

Card and currency frauds include direct attacks to steal cash or cards as well as indirect attacks to steal sensitive cardholder data that is later used to create fake cards for fraudulent withdrawals [10]. The target of these attacks is a single ATM, which may be physically manipulated for skimming, card fishing and currency trapping. Skimming is the approach to install an additional device, called a card skimmer, to capture the card's information on the magnetic strip. Lower tech card fishing and currency trapping focus on either card or cash capturing, typically using thin plates, thin metallic stripes, transparent plastic film, wires and hooks [5]. There are several security methods that deal with this threat category. Jitters, for example, vary speed and movement of cards or introduce motion. In other words, it distorts the magnetic stripe details and makes it difficult for the skimmer to read data while the card reader pulls the card into the ATM [12]. A further approach of an anti-skimming module is a jammer with the aim to disrupt a skimmer attached to the ATM dashboard. Instead of working on a mechanical level, a jammer uses an electromagnetic field to protect the cards' magnetic strips. Hence, the card reader can generate an error code that can be traced by remote monitoring tools [5].

B. Physical Attacks

Attacks that result in the physical damage of the entire ATM or a component thereof primarily focus on stealing cash from the safe [10]. But, some of these attacks are also conducted to prepare a further malicious activity on a single ATM. Vulnerable and easy targets for such attacks are off-site ATMs that are open to the public, less protected and lighter compared to bank-located machines [13]. Physical security guidelines recommend seismic detectors, magnetic contacts, alarm control panels, access control and heat sensors as alarm equipment [14]. Seismic detectors indicate abnormal vibrations and can cry havoc if an ATM is about to be raided. Heat sensors detect any form of unnatural temperature rise. Volumetric detectors on the wall can detect movements in the ATM's surrounding area. Intelligent bank note neutralization or degradation systems use bank note staining. A trigger becomes activated in case an inappropriate movement of the cassettes takes place. As a result, stolen banknotes get marked with a degradation agent or a dye.

III. LOGICAL ATM SECURITY

Logical attacks have become more sophisticated and their execution has typically been well organized [5][7][8][15]. Thus, recent examples, such as Skimer [16], Ploutus [17], Stuxnet [18] and a logical attack demonstrated at the chaos computing club congress [19] are indicators that these attacks bring up new methods and approaches to ATM crime. ATM malware is designed to steal cardholder data and PINs or to withdraw cash [9][13][15]. Typically, malware hides in the system to remain undetected as long as possible. It impairs confidentiality, integrity and authenticity of transaction data for its particular intention [5][10]. ATM networks are based on the Internet protocol and face the same attacks as other IP-related networks, e.g., denial of service (DoS), sniffing, man-in-the-middle attacks, or eavesdropping [3][10]. Communication between ATM and host can be used as entry point to launch remote attacks [5]. Even network devices like routers and switches can be targeted [3]. Logical security focuses on maintaining a secure network, protecting the OS and designing a system so that intruders cannot threaten cardholder's data and software components [5][10]. Subsequent subsections describe such measures.

A. Cardholder Data Protection

Sensitive data is the main target of logical attacks [20]. The Payment Card Industry (PCI) Data Security Standard (DSS) is for the protection of sensitive cardholder and authentication data. It proposes a set of twelve requirements divided into six areas [20]. Based on these requirements we have identified four security controls, which are needed to protect cardholder data:

- *Change control*, to guarantee that necessary and wanted changes are made only
- *Data masking*, to disguise cardholder data
- *User access control*, to restrict permsissions
- *Password policy*, to hamper password guessing

B. Host-based Firewall

To operate a secure ATM network, logical ATM security systems must be in place [5]. A firewall and a monitoring system to analyze and authenticate connection attempts are recommended in order to build such a layer of defense [5]. Instead of installing a central firewall, an integrated firewall on the ATM is feasible, controlling network communications on the processes, protocols and ports level [8].

C. Application Control

Traditional security software like antivirus software is used on desktop PCs to prevent unauthorized software execution. But, antivirus software requires processing power that often goes beyond the capabilities of an ATM and relies on a signature database that needs periodic updates. These updates can only provide protection against known malware. Consequently, malware prevention must operate within the limited resources and with a minimal "footprint" to avoid complications with ATM software [8]. Whitelisting restricts software running on an ATM to a known set of applications [8] that are tested and approved for execution. Unapproved software outside the list and malware are prohibited.

D. Full Hard Disk Encryption

Some logical attacks bypass security protection by booting the ATM from an alternative medium, such as a USB

 ISBN: 978-1-61208-438-1

stick or CD-ROM. This circumvention provides the possibility to manipulate configurations or to put malware in place [21]. As a countermeasure, the ATM hard disk can be protected with full hard disk encryption [21]. In addition, it is recommended to encrypt data on an ATM's hard disk to make it unreadable in case of theft or unauthorized access [10]. Physically protecting the hard disk is an additional safeguard, because data access becomes more difficult.

E. Patch Management

Logical security includes the handling of software vulnerabilities by patch management to ensure the efficiency and security of ATMs in a timely and efficient manner [22]. Continuous patch management provides protection against viruses, worms and known vulnerabilities within an OS [22]. An example in this context is the Slammer virus, which was responsible for network outages of different systems, such as ATMs with Windows [23]. The incident could have been prevented because Microsoft had provided a patch covering the exploited vulnerability six month before the virus spread out [23]. Needless to say, precautions have to be taken to avoid malicious misuse of update mechanisms.

F. Device-specific Requirements

Depending on the actual installation of ATMs, additional security controls are required for a higher level of defense. Examples of countermeasures include secure test utilities and device controls. Test utilities that are built in an ATM platform must be protected via access control mechanisms. Externally available devices, especially USB ports, must be controlled on BIOS or OS level.

IV. RISK ASSESSMENT

Risks must be controlled by countermeasures or safeguards [24]. Risk management is an important part of an organization's security program. It provides support in managing information security risks associated with an organization's overall mission [25]. Risk management must repeatedly be conducted in periodical time spans [26]. Each iteration begins with risk assessment [26], which is initiated at a predefined time, e.g., once a year or after a major IT change [27]. It results in the identification, estimation and prioritization of IT risks based on confidentiality, integrity and availability [24]. The result represents a temporary view that will be used for further risk management decisions [26].

A. Risk Model

The risk model specifies key terms and assessable risk factors including their relationships [24]. It defines all factors that directly or indirectly determine the severity and level of a particular risk, such as assets, threat source, threat event, likelihood, impact and countermeasure. Assets represent resources of value that need to be protected [28]. Thus, a person, physical object, organizational process or implemented technology can represent an asset. A threat is the potential for a malicious or non-malicious event that will damage or compromise an asset [28], e.g., unauthorized modification, disclosure or destruction of system components and information [24]. Depending on the degree of detail and complexity, it is possible to specify a threat as a single event, action or circumstance; or as a set of these entities [24]. A vulnerability is a weakness in the defense mechanism that can be exploited by a threat to cause harm to an asset [26][28]. This weakness can be related to security controls that either are missing or have been put in place but are somehow inefficient [24].

The likelihood of a risk consists of two aspects, i.e., the likelihood of occurrence (initiation of an attack) and the likelihood of success [24]. The likelihood of occurrence demonstrates the probability of a threat to exploit a vulnerability or a set of vulnerabilities [24]. Factors that determine this likelihood value are predisposing conditions, the presence and effectiveness of deployed countermeasures and the consideration of how certain the threat event is to occur. The likelihood of success expresses the chance that an initiated threat event will cause an adverse impact without considering the magnitude of the harm [24]. The impact describes the magnitude of expected harm on an organization [28]. To determine the impact, it is important to understand the value of the asset and the value of an undamaged system. Besides, it is advisable to consider an impact not only as a one-time loss because it can have relationships to other factors that cause consequential damage [24]. A risk is a combination of the likelihood that an identified threat will occur and the impact the threat will have on the assets under review [24]. Risk factors, such as threat, vulnerability, likelihood and impact determine the overall risk. Impact and likelihood are used to define the risk level [27].

B. Risk Assessment Process

Different risk assessment processes, frameworks and methodologies build on the same underlying process structure, which may vary in abstraction level and granularity [24][26]. These steps, which are listeted below, do not have to be strictly adhered to in sequential order. For example, it is useful to perform threat and vulnerability identification side by side to cover all risk possibilities. Also, some step iterations are necessary to get representative results [24].

1. *Definition of Assets* - No action can be taken unless it is clarified what the assets are. Asset definition seeks to identify the processes, applications and systems that are highly important and critical to an organization's daily operation [28].
2. *Identification of Threat Sources and Events* - Threat sources can be characterized based on their capability, intent and target to perform a malicious activity [24]. Once the list of sources is complete, threat events must be identified that can be initiated by a threat source. Predefined checklists are an easy way to verify whether the listed threat events can occur in the context of the assessment. But, an exclusive use of checklists can negatively influence the outcome because it may impair the free flow of creative thinking and discussing. An important step is the determination of the relevance of each threat event. If considered relevant, an event will be paired with all possible threat sources that can initiate it.

 ISBN: 978-1-61208-438-1

3. *Identification of Vulnerabilities and Predisposing Conditions* - Next, we have to identify vulnerabilities that can be exploited as well as the conditions that may increase or mitigate susceptibility. Tool support is feasible for this task. For example, vulnerability scanners automatically test internal and external system interfaces in order to find known and obvious weaknesses.
4. *Determination of Overall Likelihood* - The overall likelihood represents the probability that the threat exploits vulnerabilities against an asset [28]. To get an adequate value and to keep focused on specific aspects, the overall value is divided into likelihood of initiation/occurrence and likelihood of success. These are an assessment of the probability that a non-adversarial threat happens or an adversarial threat source launches an attack [24]. In contrast, the likelihood of success is the probability that an initiated threat event results in an adverse impact [24].
5. *Determine Magnitude of Impact* - It is necessary to determine the impact the event will have on the organization [28]. For this task, the values of reviewed assets are an important input because they show the potential harm and the severity of the impact in case of a full or partial loss. The harm can be expressed in terms of monetary, technical, operational or human impact criteria [27].
6. *Determine Risk* - The risk level is determined by combing impact and overall likelihood [24][27]. It shows the degree to which an organization is threatened [24]. Formulas, matrices or methods that are used for merging likelihood and impact must be consistent and precisely defined.

V. CASE STUDY

The aim of this case study is a risk assessment to establish a baseline assessment of risks that are faced by an ATM platform of a specific manufacturer. Thus, the risk assessment identifies all threats, vulnerabilities and impacts that cause a risk to an ATM asset. The focus on the ATM platform limits our investigation to software aspects. Thus, we mainly focus on logical risks. We'd like to mention at this point that we have to refrain from describing attacks in too much detail, because this would provide valuable information to potential attackers. However, the given information is sufficient for readers to follow the conclusions that we will draw.

A. System Characterization

The logical system structure of an ATM consists of three layers as shown in Figure 1. On the bottom end is the OS, which is on top of the hardware layer (not considered here) and builds the base for all layers above. The second layer is the ATM platform that uses the functionalities of the OS in order to communicate with hardware components. The ATM platform provides a public interface to multi-vendor ATM software and bank applications that depict the third layer. The ATM platform is designed to run on various releases of Microsoft Windows. Some of these releases are optimized for point of sale solutions, i.e., Embedded POSReady. The ATM platform implements the eXtension for Financial Services (XFS) interface specification defined in [29]. XFS does not differ between a multi-vendor ATM software and a bank application, but considers both forms of an ATM software as a Windows-based XFS application [29]. The key element of XFS is the definition of an Application Programming Interface (API) and a corresponding Service Provider Interface (SPI). The API provides access to financial services for Windows-based XFS applications. The SPI is similar to the API, but is utilized for the direct communication with vendor-specific service providers. Each of the service providers represents a peripheral device of the ATM. The XFS manager handles the overall management of the XFS subsystem. Thus, this component is responsible for establishing and mapping the communication between API and SPI.

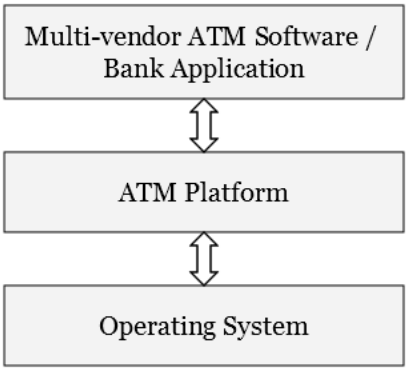


Figure 1. Logical System Structures of an ATM

B. Logical Risk Assessment

The risk assessment conducted in this case study is based on the risk assessment published in [24]. The focus of the assessment is on the ATM platform, i.e., from the ATM manufacturer's perspective. The operating system and any bank applications or other ATM software have not been considered in the evaluation (the bank's perspective).

1. *Assets* - The main assets are sensitive data, cash and the company's reputation. Cash can be more precisely defined as real cash represented by bills and coins as well as book money transferred from one bank account to another. The general term of sensitive data summarizes data and information that refers to an individual or is required to secure the system. For instance, card data, personal identification number (PIN), account data or secret keys belong to this category.
2. *Threat Sources and Events* - We have derived threat sources by interviewing ATM platform engineers and customer solutions employees. The resulting sources are: attacker (or hacker), thief, cash in transit (CIT) employee, IT specialist (in data center), bank clerk, helpdesk employee, service technician and employee of ATM manufacturer. Threat events were identified in form of brainstorming sessions. Threats were grouped to categories, which were derived from the primary objective of the threat events or an important key passage in an entire scenario:
    - *Denial of Service*, making the ATM platform unavailable to a customer by dominating some of its resources.

 ISBN: 978-1-61208-438-1

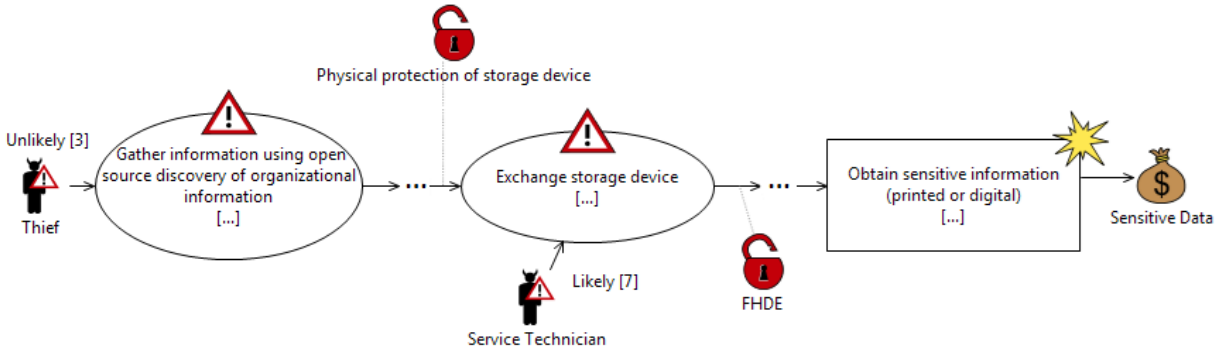


Figure 2. Snippet from Threat Diagram: *Sensitive Data Disclosure*

- *Malicious Software Injection*, injecting malicious software, such as Trojan horses, viruses or worms at the OS level or the ATM platform level.
- *Sensitive Data Disclosure*, gathering unprotected cardholder data.
- *Configuration File Modification*, changing configuration files of the ATM platform.
- *Privilege Settings Modification*, modifying configuration files, focusing on the change of the user access control model to gain more privileges.
- *Software Component Modification*, modifying an executable or an assembly of the ATM platform, assuming the adversary can decompile the target file.
- *Test Utility Exploitation*, exploiting test utilities used by service technicians, IT specialists and ATM platform engineers for maintenance.

Eventually, the events were connected to threat sources and logically ordered to create entire scenarios. As a result, a directed graph was designed for each threat group. Figure 2 shows a snippet of the graph regarding the disclosure of sensitive data. With this graphical visualization on the table, the relevance of all threat scenarios was assessed and classified as either confirmed, likely, unlikely or not applicable. This is shown in Figure 2 by a label next to the threat source.

3. *Vulnerabilities* - In order to disclose vulnerabilities in the ATM platform, we have analyzed the threat scenarios based on countermeasures recommended in Section III. For instance, as is shown in Figure 2 by the second of the two lock symbols, missing hard disk encryption may allow a thief or service technician to access and read data on an ATM's hard disk.
4. & 5. *Likelihood and Magnitude of Impact* - We have derived the likelihood of occurrence from the characteristics of particular threat sources. These characteristics had been determined in discussions with employees from the ATM manufacturer and included capabilities of threat sources as well as intent and targeting, see (24). The likelihood of success has been determined by the vulnerabilities of the ATM platform. Results of threat scenarios, which were linked to the three assets of the ATM, were assessed as very high (10) or high (8), because they caused an immediate loss when they get stolen or damaged. Harm to the ATM manufacturer is evaluated as high (8) and the impact of indirect harm is considered as moderate (5). The latter is weighted as moderate because a further threat scenario is necessary to actually cause an impact.
6. *Risk* - Risk determination has the aim to aggregate all assessed aspects of the risk factors to the risk level. We have used a likelihood impact combination matrix for that purpose, see [24]. Table I shows the distribution of threat sources for risks assigned to the seven threat groups. The numbers do not represent individual scenarios, but threat sources of such scenarios. For example, in Figure 2 we have one threat scenario with two different threat sources, i.e., thief and service technician. Table II changes the perspective and shows how countermeasures affect risks of different risk levels. The Roman numerals I to VI on the left correspond to sections III.A through III.F as well as to sections VI.A through VI.F. Thus, this table helps in identifying security controls that are useful to mitigate multiple risks at once. Similar to Table I, the numbers do not represent single threat scenarios but threat sources.

VI. DISCUSSION

The discussion about countermeasures in the literature reflects the result of the assessment in our case study. The case study additionally highlights security approaches and technologies, which were identified as most appropriate for dealing with logical ATM risks.

A. Cardholder Data Protection

We have identified change control and efficient user access control as most appropriate for protecting cardholder data and also for threat scenarios that focus on settings changes or software components of a running ATM platform. The main purpose is to guarantee that neither unnecessary nor unwanted changes are made. A change control system also supports the documentation of modifications, ensures that resources are used efficiently and services are not unnecessarily disrupted. With reference to ATMs, it can be additionally applied for ensuring PCI compliance because the change control system provides an overview of software that is deployed within the ATM environment. Although data masking is activated by default by the investigated ATM platform, there are threat sources capable to disable this

 ISBN: 978-1-61208-438-1

TABLE I. DISTRIBUTION OF RISKS

Threat Group	Risk Level				
	very high	high	mod	low	very low
Denial of Service	-	-	-	2	-
Malicious Software Injection	-	7	40	19	-
Sensitive Data Disclosure	2	8	13	-	-
Configuration File Modification	1	7	13	7	-
Privilege Settings Modification	-	1	15	14	-
Software Component Modification	1	7	37	-	-
Test Utility Exploitation	-	6	12	-	-

TABLE II. DISTRIBUTION OF COUNTERMEASURES

Countermeasure		Risk Level				
		very high	high	mod	low	very low
I	Change Control	1	7	13	7	-
	Data Masking	-	1	3	-	-
	User Access Control	-	1	15	14	-
	Password Policy	-	1	3	-	-
II	Host-based Firewall	2	6	4	1	-
III	Application Control	1	9	38	-	-
IV	Full Hard Disk Encryption	-	9	55	19	-
V	Patch Management	-	2	9	7	-
VI	Securing Test Utilities	-	4	8	-	-
	Device Control (for USB Port)	-	2	1	6	-

feature. Consequently, the approach of obfuscating data becomes inadequate if user access control is not in place. The most efficient way of implementing a user access control mechanism is by applying the user management that comes with the OS. Not a technical but an organizational countermeasure is the implementation of a password policy, which enforces a periodical change of passwords that are either used for locking user accounts or for switching to the maintenance mode of the ATM platform.

B. *Host-based Firewall*

Malicious use of the network interface can be mitigated through a host-based firewall. Such a firewall has to work on the level of protocols, ports and processes, i.e., the configuration of the firewall must specify protocols and ports that can be used by a particular process for outgoing connections. The same applies for incoming traffic. All ports and protocols that are not in use must be blocked by default.

C. *Application Control*

Protection against unauthorized software on ATMs has to focus on whitelisting, where the execution of applications and executables is limited to a predefined set. This set includes files that are required to run the OS and the ATM platform. All other executable files not in the whitelist cannot be launched, even if not malicious. As a consequence, threat scenarios that install known or tailored malware on the ATM platform fail in the execution of the malicious software. Whitelisting solutions offer a simple device control by removing the device's driver from the whitelist.

D. *Full Hard Disk Encryption*

Hard disk encryption is a powerful countermeasure against alternatively booting the system for malicious activities. Several threat events require access to an ATM's computer to boot the system from an alternative medium. Although launching an alternative OS would work because the environment is running in the RAM, access to the encrypted hard disk fails. As a result, an adversary is not able to search for sensitive data, to drop malicious files, to collect executables and dynamic link libraries from the ATM platform or to change the privileges of restricted objects. Hard disk encryption tones down threat scenarios that concentrate on stealing or exchanging a hard disk as encrypted hard disks cannot be used on another system. A Trusted Platform Module (TPM) chip, which is mounted on a computer's main board, can be used to establish this connection. Other approaches do not require additional hardware, but can compute the encryption key based on unique characteristics of installed hardware components or network location of the ATM.

E. *Patch Management*

A fundamental base for an effective patch management is appropriate hardening of a system. Compared to a firewall that works at the network side, system hardening focuses on the OS level and removes or disables all unnecessary applications, users, logins and services. For instance, non-essential applications, which may offer useful features to a user at a workstation, must be removed because they could provide a backdoor to an ATM environment. Next to hardening, a rule policy with defined user privileges must be in place. The reason is that managing a distributed system like an ATM network still provides a vector for the installation of malware by maintenance staff. Based on that groundwork, a continuous patch management allows a financial institute to provide protection against known viruses, worms and vulnerabilities within an OS.

F. *Device-specific Requirements*

For dealing with the potential danger arising from test tools used by ATM platform engineers, service technicians and IT specialists, it is important that these tools function only under certain circumstances. Especially, when the ATM

 ISBN: 978-1-61208-438-1

is in maintenance mode, the tools should support the activities on the ATM. But, in all other cases they must be disabled. Device control comes into play when the USB ports of an ATM represent possible entry points for a malicious activity. Similar to the concept of application control, device control can be implemented by whitelisting solutions too. Instead of blocking an application, a whitelisting solution can block the USB driver resulting in disabled USB ports.

VII. Related Work

DeSomer demonstrates that card skimming provides the highest ATM risk [30]. In order to detect a card skimming device or the installation of a camera for PIN capturing, the author highlights risk mitigation measures, such as jitter devices, lighting improvements or fraudulent device inhibitors. A survey about ATM security highlights that some of the security measures are obsolete and inadequate [31]. Thus, fraudulent activities can be easily performed on an ATM. The work proposes improvements in the authentication process by installing a finger vein technology or a facial recognition system. Bradbury has conducted ATM security from a logical point of view [15]. He concludes that logical fraud activities on ATMs are increasing and executed as organized and highly sophisticated attacks. Adversaries are capable to manipulate the software inside of ATMs to directly withdraw money. The severity of this issue is underlined by the fact that both banks and customers are facing heavy losses. Rasiah discusses the topic of ATM risk assessment from the same perspective as we did [32]. The author adapts a non-technical approach and investigates risk management and controls by defining general ATM security goals. The paper provides a general overview on ATM risk related topics.

VIII. Conclusion

We have discussed various aspects of ATM security, i.e., card and currency fraud, physical attacks as well as logical attacks. Logical risks of a specific ATM have been assessed in a case study to evaluate and prioritize appropriate countermeasures. The risk assessment has provided information about countermeasures in general and their importance in particular. This allows the ATM manufacturer to better plan resources for security and concentrate on the most important countermeasures first. Also, we have found out that countermeasures suggested in the literature are effective for the identified risks. By multiplying risk levels and the number of threat sources of Table II, we have identified application control, full hard disk encryption, and user access control to be most effective, as they provide protection to most identified risks. A host-based firewall is also a must for ATM security, as it protects against very high risks.

Reference

[1] B. Batiz-Lazo and R. Reid, "The Development of Cash-Dispensing Technology in the UK," IEEE Ann. Hist. Comput., vol. 33, no. 3, 2011, pp. 32–45.

[2] T. Kaltschmid, "95 percent of ATMs run on Windows XP," heise online. [Online]. Available: http://www.heise.de/newsticker/meldung/95-Prozent-aller-Geldautomaten-laufen-mit-Windows-XP-2088583.html [retrieved: 08, 2015].

[3] C. Benecke and U. Ellermann, "Securing 'Classical IP over ATM Networks'," presented at the Proceedings of the 7th conference on usenix security symposium (SSYM '98), Berkeley, CA, USA, 1998, pp. 1–11.

[4] R. T. Guerette and R. V. Clarke, "Product Life Cycles and Crime: Automated Teller Machines and Robbery," Secur. J., vol. 16, no. 1, 2003, pp. 7–18.

[5] Diebold, "ATM Fraud and Security," Diebold, 2012. [Online]. Available: http://www.diebold.com/Diebold%20Asset%20Library/dbd_atmfraudandsecurity_whitepaper.pdf [retrieved: 08, 2015].

[6] RBR, "Global ATM Market and Forecasts to 2018," Retail Bank. Res., 2013.

[7] ENISA, "ATM Crime: Overview of the European situation and golden rules on how to avoid it," 2009.

[8] GMV, "Protect your automatic teller machines against logical fraud," 2011. [Online]. Available: http://www.gmv.com/export/sites/gmv/DocumentosPDF/checker/ WhitePaper_checker.pdf [retrieved: 08, 2015].

[9] R. Munro, "Malware steals ATM accounts and PIN codes," theInquirer, 2009. [Online]. Available: http://www.theinquirer.net/inquirer/news/1184568/malware-steals-atm-accounts-pin-codes [retrieved: 08, 2015].

[10] S. Chafai, "Bank Fraud & ATM Security," InfoSec Institute, 2012. [Online]. Available: http://resources.infosecinstitute.com/bank-fraud-atm-security/ [retrieved: 08, 2015].

[11] PCI, "Information Supplement PCI PTS ATM Security Guidelines," PCI Security Standards Council, 2013. [Online]. Available: https://www.pcisecuritystandards.org/pdfs/PCI_ATM_Security_Guidelines _Info_Supplement.pdf [retrieved: 08, 2015].

[12] F. Lowe, "ATM community promotes jitter technology to combat ATM skimming," ATMMarketplace, 2010. [Online]. Available: http://www.atmmarketplace.com/article/178496/ATM community-promotes-jitter-technology-to-combat-ATM-skimming [retrieved: 08, 2015].

[13] T. Kitten, "ATM Attacks Buck the Trend," BankInfoSecurity, 2010. [Online]. Available: http://www.bankinfosecurity.com/atm-attacks-buck-trend-a-2786 [retrieved: 08, 2015].

[14] ATMSWG, "Best Practice For Physical ATM Security," ATM Security Working Group, 2009. [Online]. Available: http://www.link.co.uk/ SiteCollectionDocuments/Best_practice_for_physical_ATM_security.pdf [retrieved: 08, 2015].

[15] D. Bradbury, "A hole in the security wall: ATM hacking," Netw. Secur., vol. 2010, no. 6, 2010, pp. 12–15.

 ISBN: 978-1-61208-438-1

[16] DrWeb, "Trojan.Skimer.18 infects ATMs," Doctor Web. [Online]. Available: http://news.drweb.com/?i=4167&c=5&lng=en&p=0 [retrieved: 08, 2015].

[17] J. Leyden, "Easily picked CD-ROM drive locks let Mexican banditos nick ATM cash," BusinessWeek: Technology. [Online]. Available: http://www.theregister.co.uk/2013/10/11/mexico_atm_malware_scam/ [retrieved: 08, 2015].

[18] Metro, "Stuxnet worm 'could be used to hit ATMs and power plants,'" Metro. [Online]. Available: http://metro.co.uk/2010/11/25/stuxnet-worm-could-be-used-to-hit-atms-and-power-plants-591077/ [retrieved: 08, 2015].

[19] 30C3, "Electronic Bank Robberies - Stealing Money from ATMs with Malware," presented at the 30th Chaos Communication Congress (30C3), 2013.

[20] PCI, "PCI DSS - Requirements and Security Assessment Procedures," PCI Security Standards Council, 2013. [Online]. Available: http://de.pcisecuritystandards.org/_onelink_/pcisecurity/en2de/minisite/en/docs/PCI_DSS_v3.pdf [retrieved: 08, 2015].

[21] J. J. Leon, "The case of ATM Hard Disk Encryption," RBR Bank. Autom. Bull., vol. 318, 2013, pp. 11–11.

[22] Diebold, "Patch Management Considerations in an ATM Environment," Diebold, 2012. [Online]. Available: http://www.diebold.com/Diebold%20Asset%20Library/dbd_softwaredatamanagement_whitepaper.pdf [retrieved: 08, 2015].

[23] H. Cavusoglu, H. Cavusoglu, and J. Zhang, "Economics Of Security Patch Management," in Proceedings of the The Fifth Workshop on the Economics of Information Security (WEIS 2006), Cambridge, United Kingdom, 2006.

[24] "NIST Special Publication 800-30 Revision 1, Guide for Conducting Risk Assessments," National Institute of Standards and Technology, 2012. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/ 800-30-rev1/sp800_30_r1.pdf [retrieved: 08, 2015].

[25] G. Stoneburner, A. Y. Goguen, and A. Feringa, "SP 800-30. Risk Management Guide for Information Technology Systems," National Institute of Standards & Technology, Gaithersburg, MD, United States, 2002.

[26] R. K. Rainer, C. A. Snyder, and H. H. Carr, "Risk Analysis for Information Technology," J. Manag. Inf. Syst., vol. 8, no. 1, 1991, pp. 129–147.

[27] ENISA, "Risk Management: Implementation principles and Inventories for Risk Management/Risk Assessment methods and tools.," 2006. [Online]. Available: http://www.enisa.europa.eu/activities/risk-management/current-risk/risk-management-inventory/files/deliverables/risk-management-principles-and-inventories-for-risk-management-risk-assessment-methods-and-tools [retrieved: 08, 2015].

[28] T. R. Peltier, Information Security Fundamentals, Second Edition. Boca Raton, FL, USA: CRC Press, 2013.

[29] CEN, "Extensions for Financial Services (XFS) interface specification Release 3.20 - Part 1: Application Programming Interface (API) Service Provider Interface (SPI) Programmer's Reference.," European Committee for Standarization, 16-Apr-2014. [Online]. Available: ftp://ftp.cenorm.be/PUBLIC/CWAs/other/WS-XFS/CWA16374/CWA16374-1-2011_December.pdf [retrieved: 08, 2015].

[30] F. DeSomer, "ATM Threat and Risk Mitigation," Thai-American Business, vol. 2, 2008, pp. 28–29.

[31] F. A. Adesuyi, A. A. Solomon, Y. D. Robert, and O. I. Alabi, "A Survey of ATM Security Implementation within the Nigerian Banking Environment," J. Internet Bank. Commer., vol. 18, no. 1, 2013, pp. 1–16.

[32] D. Rasiah, "ATM Risk Management and Controls," Eur. J. Econ. Finance Adm. Sci., vol. 21, 2010, pp. 161–171.

 ISBN: 978-1-61208-438-1

Applications of Security Reference Architectures in Distributed Systems: Initial Findings of Systematic Mapping Study

Sajjad Mahmood, Muhammad Jalal Khan and Sajid Anwer
Information and Computer Science Department
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
e-mail: [smahmood, g201408880, g201303950]@kfupm.edu.sa

***Abstract*—There is an increase in use of reference architectures to support software development activities for building distributed systems. Reference architectures are helpful tools to understand and specify functionalizes of a distributed system at a higher abstraction level. From a security standpoint, a distributed system's reference architecture is one of the potential starting point to study security threats and their characteristics. Both academia and industry have proposed a number of Security Reference Architectures (SRAs), which are reference architectures specifying a conceptual model of security for a system and they provide a mechanism to specify security requirements. The main objective of this work is to investigate and better understand how security reference architecture support building secure distributed software applications. In order to meet our goal, we conducted a systematic mapping study to identify the primary studies related to SRA for distributed software development. We used customized search terms, derived from our research question, to identify literature on SRA for distributed systems. We identified that a significant number of SRAs have been developed first for defense against one or few specific types of security attacks. There is also a focus on developing SRAs to satisfy a security objective during development of distributed systems. Based on the systematic mapping study results, we suggest that there is a need to develop SRAs that help system developers simultaneously enumerate different types of security threats and systematically help to decide where we should add corresponding security patterns to mitigate them.**

Keywords-security reference architecture; reference architecture; distributed systems; systematic mapping study.

I. INTRODUCTION

The past several years have seen tremendous changes in distributed software development due to introduction of web 2.0 technologies [14], service oriented architectures [15] and cloud computing systems [16]. These distributed system development technologies have brought with them several new and complex security threats and challenges. To holistically study security of these large and complex distributed systems, we need to start our security analysis from their security reference architectures [1].

A security reference architecture is a reference architecture where security mechanisms have been added in appropriate places to provide some degree of security [1]. Furthermore, a reference architecture is an abstract system architecture that describes system functionalities without any implementation details [1]. Reference architectures are useful to specify main features of a system.

A number of SRAs have been proposed by both academia and industry vendors. For example, Chonka et al. [2] report a technique that is used to observe and discover denial of service attacks against cloud systems. Okuhara et al. [3] report Fujitsu's security architecture, which logically separates computational environments, authentication and identify management. Similarly, Oracle developed a SRA [4], which addresses data security, fraud detection and compliance with reference to their products.

The literature on SRAs provides a wealth of information on how to analyze security of a system for individual attacks, model system for a security objective(s) or how to help systems meet security compliance requirements of a government organization. For example, Bahmani et al. [5] compared different enterprise information security architecture frameworks with reference to interoperability feature. Lately, Modi et al. [6] presented a survey of intrusion detection techniques in cloud computing systems.

However, there is a lack of systematic investigation of the literature covering SRA in distributed systems. The aim of this systematic mapping study is to collate knowledge to better understand how SRAs have supported system security and identify in what ways it has been applied in the industry.

The remaining of this paper is organized as follows: Section II presents the related work. The research methodology is outlined in Section III. In Section IV, we present and discuss the initial results. Section V discusses the limitation of our study. Finally, the conclusion is presented in Section VI.

II. RELATED WORK

Security is a fundamental concern in any distributed system and a number of security reference architectures have been proposed by industry and researchers' community. Majority of security reference architectures have been proposed for a particular attack type. There has been significant focus on developing SRAs to mitigate attacks such as denial of service [22], Internet protocol spoofing and denial of service [23].

On the other hand, researchers have also focused on developing security objective specific SRA. For example, Hafner et al. [10] have used enterprise patterns to develop secure services for cloud computing systems. Lombardi and Pietro [11] used virtualization to propose an architecture for cloud protection that monitors middleware integrity.

Even though extensive research has been carried out in the security reference architecture domain, it is necessary to

 ISBN: 978-1-61208-438-1

assess the current state of research and practice, and provide practitioners with evidence that enables fostering future research directions. To the best of our knowledge, there is a lack of systematic investigation of the literature covering SRA in distributed systems.

III. RESEARCH METHODOLOGY

In order to address the research question, we applied Systematic mapping study and literature review [7] approach. A systematic mapping study and literature review is a technique to identify, analyze and interpret relevant published primary studies with reference to a specific research question. Systematic mapping studies are recommended as a review methodology [7] because they allow the researchers to systematically summarize existing evidence from literature, identify research gaps and provide a framework to position future research activities [13].

A systematic mapping study protocol consists of five main phases, as shown in Figure 1. In the first phase of our study, we formulated a research question as follows:

RQ1: How is security reference architecture supporting development of distributed systems?

Next, we constructed the search strategy in line with our research question and performed the search for relevant publications. In the third phase, the identified relevant publications were scrutinized to ensure their relevance. In the fourth phase, the selected studies were evaluated based on the quality assessment criteria. In the last phase, data was extracted from selected studies for further analysis and assessment.

A. Search Strategy

The search strategy for the systematic mapping study is based on the steps as follows:

- Derive the search terms from population, intervention and outcomes.
- Identify alternative spellings and synonyms for major terms.
- Use Boolean 'OR' and 'AND' operators.
- Verify the derived search term in major academic repositories.

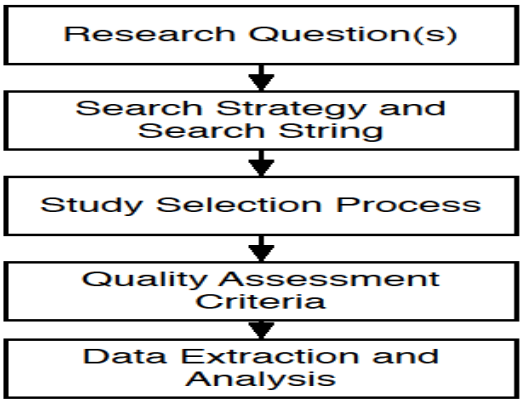


Figure 1. Systematic Mapping Study Major Process Phases.

We constructed the following search terms based on our search strategy:

- POPULATION: Distributed systems, Cloud systems, Service-oriented Architecture.
- INTERVENTION: Security Methodology.
- OUTCOME OF RELEVANCE: Different techniques to mitigate security in reference architecture, classification of SRAs.
- EXPERIMENTAL DESIGN: Systematic literature reviews and empirical studies.

We validated our search terms in major academic databases in a scoping study. The following search terms show potential relevance to the research question as follows:

- SECURITY REFERENCE ARCHITECTURE: Security Architecture OR Security Reference Architecture OR Security Design OR Security Architecture Design, Security Patterns; AND
- DISTRIBUTED SYSTEM: Distributed Systems OR Cloud Systems OR Service-oriented Architecture OR Grid Systems; AND
- TECHNIQUE: Technique OR Method OR Model OR Design.

The relevant studies retrieved through the initial search string were used as a guide for the development and validation of the final search string. In the scoping study, we used some relevant publications, which we had previously identified to cross check the validity of the search terms. A broad search was conducted between February 2015 and May 2015 to identify relevant articles published (or available on-line) up to May 2015.

B. Publication Selection

The following inclusion criteria were used:

- Peer-reviewed studies.
- Papers focus on answering our research question.
- Papers published in English.

We applied the exclusion criteria as follows:

- Papers that are not published in English.
- Papers with no link with the research question.
- Grey publications, that is, papers without bibliographic information.
- In the case of duplicate papers, the most complete version published.

Next, each paper was evaluated against the quality assessment criteria shown in Table 1. Each quality assessment criterion has two answers: 'Yes' or 'No' with scores of '1' and '0', respectively. The sum of the quality criteria resulted in the quality score for a particular paper. In this study, we only consider publications with a quality score greater than 75%. As a result, 58 papers were finally selected, which met the inclusion and quality assessment criteria.

 ISBN: 978-1-61208-438-1

TABLE I. QUALITY ASSESSMENT

Quality Criteria	Possible Answers
Is there a rationale for why the study was undertaken? [8]	Yes =1 No =0
Are the research goals clearly stated?	Yes =1 No =0
Is the proposed technique clearly described?	Yes =1 No =0
Is the research empirically validated?	Yes =1 No =0
Are the limitations of this study explicitly discussed? [9]	Yes =1 No =0
Is the study supported by a tool?	Yes =1 No =0

Initially, when synthesizing the data, data was extracted from the final selection of papers as follows: study details, study research methodology, assessment details and study findings.

IV. RESULTS AND DISCUSSION

The total number of results retrieved using the search terms in the electronic databases are shown in Table 2. After the initial round of screening by reading the title and abstract, seventy one studies belonging to different electronic research databases were selected. After full text readings in the second screening and quality assessment, 58 primary studies were finally selected. Figure 2 shows temporal view of the selected articles from the systematic review, sorted by year of publication. Appendix A presents the primary studies in the review.

TABLE II. SEARCH EXECUTION

Resource	Total Results	Initial Selection	Final Selection
ACM	200	11	10
IEEE Xplore	427	36	27
Science Direct	350	18	15
Springer	61	6	6
Total	1038	71	58

To answer the research question, the data was carefully extracted and synthesized from the 58 finally selected primary studies. We classified SRAs into four main categories as shown in Table 3.

In our study, the most highly cited category is 'attack specific SRA' (60%). Distributed system infrastructure uses virtualization techniques and provide their services through standard internet protocols [6]. Distributed systems are vulnerable to traditional security attacks such as Internet protocol spoofing, routing information protocol attacks, denial of service attacks, etc. Hence, there has been a significant focus on developing SRAs to incorporate specific attack detection and prevention mechanisms in distributed system infrastructure to mitigate security attacks. Table 4 shows a list of popular types of attacks addressed by researchers and industry practitioners.

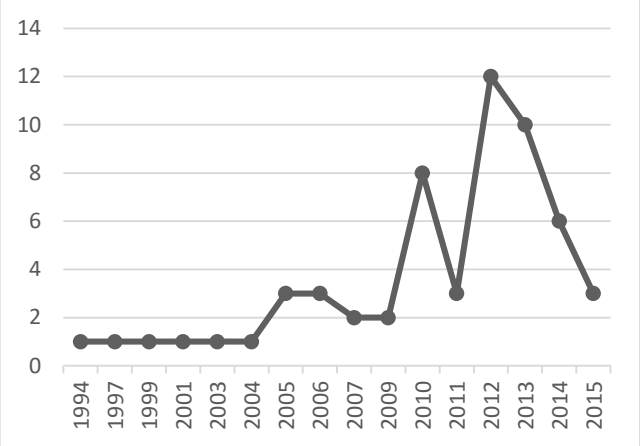


Figure 2. Temporal view of studies

'Security objective specific SRA' is the second highly cited category (reported by 34 % of the articles selected from the systematic mapping study and review). Researchers have also considered developing SRAs oriented to some specific security objectives. For example, Hafner et al. [10] have used enterprise patterns to develop secure services for cloud computing systems. Lombardi and Pietro [11] used virtualization to propose an architecture for cloud protection that monitors middleware integrity. Lately, Fernandez et al. [1] presented a method to build a SRA for cloud systems using security patterns and misuse patterns.

TABLE III. LIST OF SRAS CATEGORIES

Categories	Studies	Count	%
Attack Specific SRA	A1, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A15, A17, A18, A21, A23, A24, A25, A27, A28, A30, A37, A39, A40, A41, A42, A46, A47, A49, A53, A54, A55, A57, A58	35	60.30
Security Objective Specific SRA	A2, A14, A19, A22, A29, A31, A32, A33, A34, A35, A36, A38, A43, A44, A45, A48, A50, A51, A52, A56	20	34.48
Industry Specific SRA	A21, A26	2	3.44
Vendor Specific SRA	A16	1	1.72

Furthermore, less frequently cited categories are 'industry specific SRA' and 'vendor specific SRA. There has been couple of SRAs developed for a specific industry. For example, Cohen [12] developed a SRA for industrial control systems. Similarly, Bahmani et al. [5] discussed five enterprise security reference architectures, namely, Gartner

 ISBN: 978-1-61208-438-1

framework [17], SABSA [18], roadmap for information security across the enterprise framework [19], agile governance model based model [20] and intelligent service-oriented enterprise security architecture [21].

TABLE IV. LIST OF POPULAR TYPES OF ATTACKS

Attack Type	%
Authentication/Authorization	45
Denial of Service	22
Injection	20
Denial of Service	20
Man in the Middle Attack	14
Data Tempering	11
Brute Force	5

It is important to note that over the years, a significant number of SRAs have been developed by industry vendors. All major industrial vendors like IBM, Microsoft, Oracle, Cisco, VMware and Amazon have developed SRAs for their product range. However, in our study, we have not included them as primary studies because most of vendor specific SRAs are available in form of white papers, which do not satisfy our inclusion criteria, as mentioned in Section 2. Hence, we have included only on primary study regarding Fujistu's SRA reported by Okuhara et al. [3].

V. LIMITATIONS

Similarly to any systematic mapping study and literature review, our results also depend on the used keywords and the limitations of the search engines. In order to limit the risk of incompleteness in keywords lists, we used alternative spellings and synonyms to build the search terms.

Application of inclusion, exclusion criteria and primary study selection process are also subject to threats to validity of the study. In order to mitigate this threat, all systematic mapping study phases were carried out iteratively with continuous feedback from authors of the paper.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we conducted a systematic mapping study to investigate the use of SRA to support development of distributed systems. Fifty eight studies were finally included, which were further classified into four categories, namely, 'attack specific SRA', 'security objective specific SRA', 'industry specific SRA' and 'vendor specific SRA'.

Through this systematic mapping study, we identified that researchers have mainly focused on developing SRAs for one or group of individual security attacks. There also has been a focus on developing SRAs oriented to security objectives such as monitoring data and using security patterns to add security mechanisms at appropriate components of a system.

We believe that the results presented in our systematic mapping study and review can be useful for software engineering community as it provides an initial body of knowledge regarding SRAs. In the future, we intend to expand this systematic review to further analyze individual categories and discuss highly cited types of attacks and security objectives in the literature. Another area for future work is to empirically study different industry vendor SRAs and their impact on improving security of distributed systems.

ACKNOWLEDGMENT

The authors would like to thank King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia for continuous support in research. This research is supported by the Deanship of Scientific Research at KFUPM under research grant IN131013.

REFERENCES

[1] E. B. Fernandez, R. Monge, and K. Hashizume, "Building a security reference architecture for cloud systems," Requirements Engineering, 2015, pp. 1-25.

[2] A. Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," Journal of Network and Computer Applications, vol. 34, 2011, pp. 1097-1107.

[3] M. Okuhara, T. Shiozaki, and T. Suzuki, "Security architecture for cloud computing," Fujitsu Sci. Tech. J, vol. 46, 2010, pp. 397-402.

[4] M. Wilkins, "Orcale Reference Architecture: Cloud Foundation Architecture " Technical Report E24529-01 - Orcale Corporation, 2011.

[5] F. Bahmani, M. Shariati, and F. Shams, "A survey of interoperability in Enterprise Information Security Architecture frameworks," in Information Science and Engineering (ICISE), 2010 2nd International Conference on, 2010, pp. 1794-1797.

[6] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A Survey of Instrusion Detection Techniques in Cloud," Journal of Network and Computer Applications, vol. 36, 2013, pp. 42-57.

[7] B. Kitchenham and C. Charters, "Guidelines for Performing Systematic Literarature Reviews in Software Engineering," Keele University and Durham University Joint Report, 2007.

[8] S. Mahdavi-Hezavehi, M. Galster, and P. Avgeriou, "Variability in quality attributes of service-based software systems: A systematic literature review," Information and Software Technology, vol. 55, 2013, pp. 320-343.

[9] W. Ding, P. Liang, A. Tang, and H. Van Vliet, "Knowledge-based approaches in software documentation: A systematic literature review," Information and Software Technology, vol. 56, 2014, pp. 545-567.

[10] M. Hafner, M. Memon, and R. Breu, "SeAAS - A Reference Architecture for Security Services in SOA," Journal of Universal Computer Science, vol. 15, 2009, pp. 2916-2936.

[11] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing," Journal of Network and Computer Applications, vol. 34, 2011, pp. 1113-1122.

[12] F. Cohen, "A Reference Architecture Approach to ICS Security," presented at the 4th International Symposium on Resilient Control Systems, 2011, pp. 9-11.

[13] J. M. Verner, O. P. Brereton B. A. Kitchenham, M. Turner, and M. Niazi, "Systematic Literature Reviews in Global Software Development: A Tertiary Study" in proceedings of the 16th International Conference on Evaluation and Assessment in Software Engineering, 2012, pp. 2-11.

[14] U. Sivarajah, Z. Irani, and S. Jones, "Application of Web 2.0 Technologies in E-Government: A United Kingdom Case Study", in proceedings of 7th Hawaii Inernational Conference on System Sciences, 2014, pp. 2221-2230.

[15] D. Krafzig, K. Banke, and D. Slama, "Enterprise SOA: Service-Oriented Architecture Best Practices", Pearson Education, 2005.

[16] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications and Approaches", Wireless Communications and Mobile Computing, vol. 13, 2013, pp. 157-1611.

 ISBN: 978-1-61208-438-1

[17] T. Scholtz, "Structure and Content of an Enterprise Information Security Architecture", Gartner, 2006.

[18] J. Sherwood, A. Clark, and D. Lynas, "Enterprise Security Architecture: A Business-Driven Approach", CMP Books, 2015.

[19] J. A. Aderson and V. Rachamadugu, "Managing Security and Privacy Integration Across Enterprise Business Process and Infrastrucutre", in proceedings of IEEE International Conference on Service Computing, 2008, pp. 351-358.

[20] J.J. Korhonen, M. Yildiz, and J. Mykkanen, "Governance of Information Security Elements in Service-Oriented Enterprise Architecture in Pervasive Systems", in proceedings of 10th International Symposium on Algorithms and Networks, 2009, pp. 768-773.

[21] J. Sun and Y. Chen, "Intelligent Enterprise Information Security Architecture Based on Service Oriented Architecture", in proceedings of International Semiar on Future Information Technology and Management Engineering, 2008, pp. 196-200.

[22] W. Itani and A. Kayssi, "SPECSA: a scalable, policy-driven, extensible, and customizable security architecture for wireless enterprise applications," Computer Communications, vol. 27, 2004, pp. 1825-1839.

[23] G. Yang et al., "Analysis of security threats and vulnerability for cyber-physical systems," in proceeding of 3rd International Conference on Computer Science and Network Technology (ICCSNT), 2013, pp. 50-55.

APPENDIX A: SYSTEMATIC MAPPING STUDY PRIMARY STUDIES

A1: S. Muftic and M. Sloman, "Security architecture for distributed systems," Computer Communications, vol. 17, pp. 492-500, 1994.

A2: M. Moriconi, Q. Xiaolei, R. A. Riemenschneider, and G. Li, "Secure software architectures," in Security and Privacy, 1997.

A3: R. Molva, "Internet security architecture," Computer Networks, vol. 31, pp. 787-804, 1999.

A4: M. S. Olivier, "Towards a configurable security architecture," Data and Knowledge Engineering, vol. 38, pp. 121-145, 2001.

A5: V. Varadharajan and D. Foster, "A Security Architecture for Mobile Agent Based Applications," World Wide Web, vol. 6, pp. 93-122, 2003/03/01 2003.

A6: W. Itani and A. Kayssi, "SPECSA: a scalable, policy-driven, extensible, and customizable security architecture for wireless enterprise applications," Computer Communications, vol. 27, pp. 1825-1839, 2004.

A7: M. Debbabi, M. Saleh, C. Talhi, and S. Zhioua, "Security Analysis of Mobile Java," in Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on, 2005, pp. 231-235.

A8: D. Gabrijelčič, B. J. Blažič, and J. Tasič, "Future active Ip networks security architecture," Computer Communications, vol. 28, pp. 688-701, 2005.

A9: G. Gousios, E. Aivaloglou, and S. Gritzalis, "Distributed component architectures security issues," Computer Standards & Interfaces, vol. 27, pp. 269-284, 2005.

A10: A. Vorobiev and J. Han, "Secrobat: Secure and Robust Component-based Architectures," in Software Engineering Conference, 2006. APSEC 2006. 13th Asia Pacific, 2006, pp. 3-10.

Proceedings., 1997 IEEE Symposium on, 1997, pp. 84-93.

A11: C. Lu, T. Zhang, W. Shi, and H.-H. S. Lee, "M-TREE: A high efficiency security architecture for protecting integrity and privacy of software," Journal of Parallel and Distributed Computing, vol. 66, pp. 1116-1128, 2006.

A12: T. Fægri and S. Hallsteinsen, "A Software Product Line Reference Architecture for Security," in Software Product Lines, T. Käköla and J. Duenas, Eds., ed: Springer Berlin Heidelberg, 2006, pp. 275-326.

A13: C. Martin and K. A. Abuosba, "Utilizing a Service Oriented Architecture for Information Security Evaluation and Quantification," in Business-Driven IT Management, 2007. BDIM '07. 2nd IEEE/IFIP International Workshop on, 2007, pp. 114-115.

A14: V. S. Sharma and K. S. Trivedi, "Quantifying software performance, reliability and security: An architecture-based approach," Journal of Systems and Software, vol. 80, pp. 493-509, 2007.

A15: R. Shioya, K. Daewung, K. Horio, M. Goshima, and S. Sakai, "Low-Overhead Architecture for Security Tag," in Dependable Computing, 2009. PRDC '09. 15th IEEE Pacific Rim International Symposium on, 2009, pp. 135-142.

A16: M. Okuhara, T. Shiozaki, and T. Suzuki, "Security architecture for cloud computing," Fujitsu Sci. Tech. J, vol. 46, pp. 397-402, 2010.

A17: J. Li, X. Lu, and G. Gao, "A mobile ad hoc network security architecture based on immune agents," in Communication Systems, Networks and Applications (ICCSNA), 2010 Second International Conference on, 2010, pp. 224-227.

A18: S. Donglai, W. Yue, W. Tian, L. Yang, L. Ning, and T. Junhua, "Design and Construction of a Prototype Secure Wireless Mesh Network Testbed," in Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on, 2010, pp. 345-350.

A19: R. G. Addie and A. Colman, "Five Criteria for Web-Services Security Architecture," in Network and System Security (NSS), 2010 4th International Conference on, 2010, pp. 521-526.

A20: H. Xu, F. Wan, H. Zheng, and M. Xu, "A Security Architecture Model of CSCW System," in Management and Service Science (MASS), 2010 International Conference on, 2010, pp. 1-4.

A21: F. Bahmani, M. Shariati, and F. Shams, "A survey of interoperability in Enterprise Information Security Architecture frameworks," in Information Science and Engineering (ICISE), 2010 2nd International Conference on, 2010, pp. 1794-1797.

A22: M. Asgarnezhad, R. Nasiri, and S. Sahebhonar, "Analysis and Evaluation of Two Security Services in SOA," in Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on, 2010, pp. 562-568.

A23: A. M. Rossudowski, H. S. Venter, J. H. P. Eloff, and D. G. Kourie, "A security privacy aware architecture and protocol for a single smart card used for multiple services," Computers & Security, vol. 29, pp. 393-409, 2010.

A24: G. Dini and I. Savino, "A Security Architecture for Reconfigurable Networked Embedded Systems," International Journal of Wireless Information Networks, vol. 17, pp. 11-25, 2010/06/01 2010.

A25: T. Yuan, S. Biao, and H. Eui-Nam, "Towards the Development of Personal Cloud Computing for Mobile Thin-Clients," in Information Science and Applications (ICISA), 2011 International Conference on, 2011, pp. 1-5.

A26: F. Cohen, "A reference architecture approach to ICS security," in Resilient Control Systems (ISRCS), 2011 4th International Symposium on, 2011, pp. 21-25.

A27: L. Xiaoli, C. Jinhua, and L. Min, "A Simple Security Model Based on Cloud Reference Model," in Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2011 Tenth International Symposium on, 2011, pp. 155-159.

A28: H. Abie and I. Balasingham, "Risk-based adaptive security for smart IoT in eHealth," presented at the Proceedings of the 7th International Conference on Body Area Networks, Oslo, Norway, 2012.

A29: D. Allam, "A unified formal model for service oriented architecture to enforce security contracts," presented at the Proceedings of the 11th annual international conference on Aspect-oriented Software Development Companion, Potsdam, Germany, 2012.

A30: A. Sharma, V. Fusenig, I. Schoen, and A. Kannan, "Bridging the security drawbacks of virtualized network resource provisioning model," presented at the Proceedings of the 1st European Workshop on Dependable Cloud Computing, Sibiu, Romania, 2012.

A31: S. Rangarajan, M. Verma, A. Kannan, A. Sharma, and I. Schoen, "V2C: a secure vehicle to cloud framework for virtualized and on-demand service provisioning," presented at the Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Chennai, India, 2012.

 ISBN: 978-1-61208-438-1

A32: Y. F. Wang, W. M. Lin, T. Zhang, and Y. Y. Ma, "Research on application and security protection of Internet of Things in Smart Grid," in Information Science and Control Engineering 2012 (ICISCE 2012), IET International Conference on, 2012, pp. 1-5.

A33: G. Mathew, "Security considerations and reference architecture of a cyber computing infrastructure for online education," in E-Learning, E-Management and E-Services (IS3e), 2012 IEEE Symposium on, 2012, pp. 1-6.

A34: Y. Chenghua, Z. Qi, and Z. Zhiming, "Study on Information Security Assurance Architecture in Internet," in Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on, 2012, pp. 293-296.

A35: J. Montelibano and A. Moore, "Insider Threat Security Reference Architecture," in System Science (HICSS), 2012 45th Hawaii International Conference on, 2012, pp. 2412-2421.

A36: T. Okubo, H. Kaiya, and N. Yoshioka, "Mutual Refinement of Security Requirements and Architecture Using Twin Peaks Model," in Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, 2012, pp. 367-372.

A37: L. Lan, "Study on security architecture in the Internet of Things," in Measurement, Information and Control (MIC), 2012 International Conference on, 2012, pp. 374-377.

A38: J. Li, B. Li, T. Wo, C. Hu, J. Huai, L. Liu, and K. P. Lam, "CyberGuarder: A virtualization security assurance architecture for green cloud computing," Future Generation Computer Systems, vol. 28, pp. 379-390, 2012.

A39: A. Talib, R. Atan, R. Abdullah, and M. Murad, "Ensuring Security and Availability of Cloud Data Storage Using Multi Agent System Architecture," in Knowledge Technology. vol. 295, D. Lukose, A. Ahmad, and A. Suliman, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 343-347.

A40: W. Scacchi and T. A. Alspaugh, "Processes in securing open architecture software systems," presented at the Proceedings of the 2013 International Conference on Software and System Process, San Francisco, CA, USA, 2013.

A41: M. Shtern, B. Simmons, M. Smit, and M. Litoiu, "An architecture for overlaying private clouds on public providers," presented at the Proceedings of the 8th International Conference on Network and Service Management, Las Vegas, Nevada, 2013.

A42: M. Almorsy, J. Grundy, and A. S. Ibrahim, "Automated software architecture security risk analysis using formalized signatures," presented at the Proceedings of the 2013 International Conference on Software Engineering, San Francisco, CA, USA, 2013.

A43: A. Guerrieri, L. Geretti, G. Fortino, and A. Abramo, "A service-oriented gateway for remote monitoring of building sensor networks," in Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2013 IEEE 18th International Workshop on, 2013, pp. 139-143.

A44: W. Ruoyu, Z. Xinwen, A. Gail-Joon, H. Sharifi, and X. Haiyong, "ACaaS: Access Control as a Service for IaaS Cloud," in Social Computing (SocialCom), 2013 International Conference on, 2013, pp. 423-428.

A45: G. Yang, P. Yong, X. Feng, Z. Wei, W. Dejin, H. Xuefeng, L. Tianbo, and L. Zhao, "Analysis of security threats and vulnerability for cyber-physical systems," in Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on, 2013, pp. 50-55.

A46: A. Masood, "Cyber security for service oriented architectures in a Web 2.0 world: An overview of SOA vulnerabilities in financial services," in Technologies for Homeland Security (HST), 2013 IEEE International Conference on, 2013, pp. 1-6.

A47: D. Gros, M. Blanc, J. Briffaut, and C. Toinard, "PIGA-cluster: A distributed architecture integrating a shared and resilient reference monitor to enforce mandatory access control in the HPC environment," in High Performance Computing and Simulation (HPCS), 2013 International Conference on, 2013, pp. 273-280.

A48: L. Zhang, Q. Wang, and B. Tian, "Security threats and measures for the cyber-physical systems," The Journal of China Universities of Posts and Telecommunications, vol. 20, Supplement 1, pp. 25-29, 2013.

A49: A. De Santis, A. Castiglione, U. Fiore, and F. Palmieri, "An intelligent security architecture for distributed firewalling environments," Journal of Ambient Intelligence and Humanized Computing, vol. 4, pp. 223-234, 2013/04/01 2013.

A50: E. B. Fernandez, R. Monge, and K. Hashizume, "Building a security reference architecture for cloud systems," Requirements Engineering, pp. 1-25, 2015.

A51: O. E. C, E. B. Fernandez, Ra, #250, and l. M. A, "Towards Secure Inter-Cloud Architectures," presented at the Proceedings of the 8th Nordic Conference on Pattern Languages of Programs (VikingPLoP), Vihula, Estonia, 2014.

A52: Y. Tonghao and Y. Bin, "Study of cryptography-based cyberspace data security," in Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on, 2014, pp. 1-7.

A53: M. Jouini, L. B. A. Rabai, and A. B. Aissa, "Classification of Security Threats in Information Systems," Procedia Computer Science, vol. 32, pp. 489-496, 2014.

A54: H. Suleiman, I. Alqassem, A. Diabat, E. Arnautovic, and D. Svetinovic, "Integrated smart grid systems security threat model," Information Systems, 2014.

A55: S. Sicari, C. Cappiello, F. De Pellegrini, D. Miorandi, and A. Coen-Porisini, "A security-and quality-aware system architecture for Internet of Things," Information Systems Frontiers, pp. 1-13, 2014/11/04 2014.

A56: A. K. Dwivedi and S. K. Rath, "Incorporating Security Features in Service-Oriented Architecture using Security Patterns," SIGSOFT Softw. Eng. Notes, vol. 40, pp. 1-6, 2015.

A57: J. Maerien, S. Michiels, D. Hughes, C. Huygens, and W. Joosen, "SecLooCI: A comprehensive security middleware architecture for shared wireless sensor networks," Ad Hoc Networks, vol. 25, Part A, pp. 141-169, 2015.

A58: P. Karpati, A. L. Opdahl, and G. Sindre, "Investigating security threats in architectural context: Experimental evaluations of misuse case maps," Journal of Systems and Software, vol. 104, pp. 90-111, 2015.

 ISBN: 978-1-61208-438-1

Cif: A Static Decentralized Label Model (DLM) Analyzer to Assure Correct Information Flow in C

Kevin Müller
and Sascha Uhrig
Airbus Group Innovations
Munich, Germany
Email: Kevin.Mueller@airbus.com
Sascha.Uhrig@airbus.com

Michael Paulitsch
Thales Austria GmbH
Vienna, Austria
Email: Michael.Paulitsch@thalesgroup.com

Georg Sigl
Technische Universität München
Munich, Germany
Email: sigl@tum.de

***Abstract*—For safety-critical and security-critical Cyber-Physical Systems in the domains of aviation, transportation, automotive, medical applications and industrial control correct software implementation with a domain-specific level of assurance is mandatory. Particularly in the aviation domain, the evidence of reliable operation demands new technologies to convince authorities of the proper implementation of avionic systems with increasing complexity. Two decades ago, Andrew Myers developed the Decentralized Label Model (DLM) to model and prove correct information flows in applications' source code. Unfortunately, the proposed DLM targets Java applications and is not applicable for today's avionic systems. Reasons are issues with the dynamic character of object-oriented programming or the in general uncertain behaviors of features like garbage collectors of the commonly necessary runtime environments. Hence, highly safety-critical avionics are usually implemented in C. Thus, we adjusted the DLM to the programming language C and developed a suitable tool checker, called *Cif*. Apart from proving the correctness of the information flow statically, Cif is also able to create a dependency graph to represent the implemented information flow graphically. Even though this paper focuses on the avionic domain, our approach can be applied equally well to any other safety-critical or security-critical system.**

This paper demonstrates the power of Cif and its capability to graphically illustrate information flows, and discusses its utility on selected C code examples.

Keywords–Security, High-Assurance, Information Flow, Decentralized Label Model

I. INTRODUCTION

In the domain of aviation, software [1] and hardware [2] development has to follow strict development processes and requires certification by aviation authorities. Recently developers of avionics, the electronics on-board of aircrafts, have implemented systems following the concepts of Integrated Modular Avionics (IMA) [3] to reduce costs. For security aspects, there are recent research activities in the topic of Multiple Independent Levels of Security (MILS) [4][5]. Apart from having such architectural approaches to handle the emerging safety and security requirements for mixed-criticality systems, the developers also need to prove the correct implementation of their software applications. For safety, the aviation industry applies various forms of code analysis [6][7][8] in order to evidently ensure correct implementation of requirements. For security, in particular secure information flow, the aviation industry only has limited means available, which are not mandatory yet. Here, the Decentralized Label Model (DLM) [9] that was developed two decades ago, is a promising approach as it is able to prove correct information flows according to a defined flow policy by introducing annotations into the source code. These annotations allow the modeling of the information flow policy directly on source code level avoiding additional translations between model and implementation. In short, DLM extends the type system of a programming language and ensures that the defined information flow policy using label annotations of variables is not violated in the program flow.

DLM is currently available only for Java [10]. Hence, our research challenge is to adapt this model to the C programming language for being able to use it for highly critical avionic applications. We believe annotated source code allowing to check the information flow against the formally proven DLM will help to achieve future security certifications following the framework of Common Criteria with assurance levels beyond EAL 4 [11][12][13] or equivalent avionics security levels. In this paper, we focus our contributions 1) on demonstrating the powerful features of our DLM instantiation for the C language called *Cif*, 2) including the ability of graphically represent the information flows in C programs, and 3) on presenting and discussion common use case examples presented in C code snippets. The instantiation of DLM to C extends its field of use to verify information flow properties to high-assurance embedded systems. The great importance of this research can only be acknowledged if safety software development of aerospace systems and its (in-)ability to use Java has been fully understood. Java is a relatively strongly typed language and, hence, appears at first sight as a very good choice. Among others, the dynamic character of object-oriented languages such as Java introduces additional issues for the certification process [14]. Furthermore, common features such as the Java Runtime Environment introduces potentially unpredictable and harmful delays during execution, which are not acceptable in high-criticality applications requiring high availability and real-time properties like low response times (e.g., avionics).

The remainder of this paper is organized as follow: Section II discusses recent research papers fitting to the topic of this paper. In Section III, we introduce the DLM as described by Myers initially. Our adaptation of DLM to the C language and the resulting tool checker *Cif* are described in Section IV. In Section V, we discuss common code snippets and their verification using *Cif*. This also includes the demonstration

 ISBN: 978-1-61208-438-1

of the graphical information flow output of our tool. Finally, we conclude our work in Section VI.

II. Related Work

Sabelfeld and Myers present in [15] an extensive survey on research of security typed languages within the last decades. The content of the entire paper provides a good overview to frame the research contribution of our paper. Myers and Liskov present in [9] their ideas of the decentralized trust relation in program information flows called DLM. The authors instantiated DLM to the programming language Java. Known applications (appearing to be of mostly academic nature) using Jif as verification method are:

- *Civitas*: a secure voting system
- *JPmail*: an email client with information-flow control
- *Fabric, SIF* and *Swift*: being web applications.

In this paper, DLM is adapted to the programming language C for extending the field of use to high-assurance embedded systems. In [16] Nielson et al. present their research work on the application of DLM and propose improvements to the model that have been identified as useful during the application activities. Both research groups, Nielson's one and the author's one, have been in close exchange in the recent years, particularly discussing the application of DLM to C and discovering related use cases.

Greve proposed in [17] the Data Flow Logic (DFL). This C language extension augments source code and adds security domains to variables. Furthermore, flow contracts between domains can be formulated. These annotations describe an information flow policy, which can be analysed by a DFL prover. DFL has been used to annotate the source code of a Xen-based Separation Kernel [18]. Compared to this approach of using mandatory access control, we used a decentralized approach for assuring correct information flow in this paper. The decentralized approach introduces a mutual distrust among data owners, all having an equal security level. Hence, DLM avoids the automatically given hierarchy of the approaches of mandartory access control usually relying on at least one super user.

Khedker et al. published a book [19] on several theoretical and practical aspects of data flow analysis. However, Khedker does not mention DLM as technology. Hence, the DLM research extends his valuable contributions.

III. Decentralized Label Model (DLM)

A. General Model

The DLM [9] is a language-based technology allowing to prove correct information flows within a program. The model uses *principals* to express flow policies. By default a mutual distrust is present between all defined principals. Principals can delegate their authority to other principals and, hence, can issue a trust relation. In DLM, principals own data. On this data they define read (confidentiality) and write (integrity) policies for other principals in order to grant access to it. Confidentiality policies are expressed by `owners->readers`. Integrity policies use the syntax: `owners<-writers`. The union of owners and readers or writers respectively defines the effective set of readers or writers of a data item. DLM offers two special principals:

1) Top Principal $*$: As owner representing the set of all principals; as reader or writer representing the empty set of principals, i.e. effectively no other principal except the involved owners of this policy
2) Bottom Principal _: As owner representing the empty set of principals; as reader or writer representing the set of all principals.

Additional information on this are provided in [20]. In practice, DLM policies are expressed by *labels* that annotate variables in the source code. An example is:

```
int {Alice->Bob; Alice<-_} x;
int {*->_; *<-*} y;
```

Listing 1. Declaration of a DLM-annotated Variable

This presents a label definition using curly braces as *token*. The remainder will use the compiler technology-based term *token* and the DLM-based term *annotation* as synonyms. In the example of Listing 1, the principal `Alice` owns the data stored in the integer variable `x` for both the confidentiality and integrity policy. The first part of the label `Alice->Bob` expresses the confidentiality or readers policies. In this example, the owner Alice allows Bob to read the data. The second part of the label `Alice<-_` defines that Alice allows all other principals write access to the variable `x`. For the declaration of `y`, the reader policy expresses that all principals believe that all principals can read the data and the writer policy expresses that all principals believe that no principal has modified the data. Overall, this variable has low flow restrictions.

In DLM, one may also form a conjunction of principals, like `Alice&Bob->Chuck`. This confidentiality policy is equivalent to `Alice->Chuck;Bob->Chuck` and means that both, the beliefs of Alice and Bob, have to be fulfilled [21].

B. Information Flow Control

Using these augmentations on a piece of source code, a static checking tool is able to prove whether all beliefs expressed by labels are fulfilled. A *correct* information flow is allowed if data flows from a source to an *at least equally restricted* destination. In contrast, an invalid flow is detected if data flows from a source to a destination that is less restricted than the source. A destination is *at least as restricted* as the source if:

- the confidentiality policy keeps or increases the set of owners and/or keeps or decreases the set of readers, and
- the integrity policy keeps or decreases the set of owners and/or keeps or increases the set of writers

Listing 2 shows an example of a valid direct information flow from the source variable x to the destination y. Apart from these direct assignments, DLM is also able to detect invalid implicit flows. The example in Listing 3 causes an influence on variable x if the condition $y == 0$ is true. Hence, depending on the value of y the data in variable x gets modified, i.e., by observing the status of x it is possible to retrieve the value of y. However, y is more restrictive than x, i.e., x is not allowed by the defined policy to observe the value of y. Thus, the flow in Listing 3 is invalid.

 ISBN: 978-1-61208-438-1

```
int {Alice->Bob; Alice<-_} x = 1;
int {Alice&Bob->*; Alice<-_}  y = 0;

y = x;
```

Listing 2. Valid Direct Information Flow

```
int {Alice->Bob; Alice<-_} x = 1;
int {Alice&Bob->*; Alice<-_} y = 0;

if(y == 0)
  x = 0;
```

Listing 3. Invalid Implicit Information Flow

To analyse those implicit flows, DLM also examines each instruction against the current label of the Program Counter (PC). Using Jif as template, the PC represents the current context in the program and not the actual program counter register [22]. A statement is only valid if the PC is *no more restrictive* than the involved variables of the statement. The PC label is calculated for each program block and re-calculated at its entrance depending on the condition the block has been entered.

IV. Decentralized Label Model (DLM) for C Language (Cif)

A. Extending the C Language with DLM Annotations

1) Type Checking Tool: The first step of our work was to define C annotations in order to apply DLM to this language. An annotated C program shall act as input for the DLM checker, in the following called *C Information Flow (Cif)*. Cif analyzes the program according to the defined information flow policy. Depending on the syntax of the annotations, the resulting C code can no longer be used as input for usual C compilers, such as the *gcc*. To still be able to compile the program, three major possibilities for implementing the Cif are available:

1) a Cif checking tool that translates the annotated input source code into valid C code by removing all labels
2) a DLM extension to available compilers, such as *gcc*
3) embedding labels into compiler-transparent comments using */* label */*

We decided for Option 1. We did not consider Option 2 to avoid necessary coding efforts for modifying and maintaining a special C compiler. We also did not take Option 3, due to the higher error-proneness resulting from the fact that our checker, additionally, had to decide whether a comment's content is a label or a comment. If a developer does not comply with the recognition syntax for labels, the checker could interpret actual labels as comments and omit their analysis. In worst-case the checker indicates that a program's information flow is correct without verification of labels. Hence, it would introduce the risk of false-positives.

For being able to analyze the C source code statically, the first step in the tool chain is to resolve all macro definitions and to include the header files into one file. Fortunately, this step can be performed by using the *gcc*, since the compiler does not perform a syntax verification during the macro replacement. The resulting file then is used as input for our Cif checking tool. If Cif does not report any information flow violation, the tool will create a C-compliant source code by removing all annotations. This plain C source file can be used as input file for further source code verifications, e.g., by Astrée [7], or as input for the compilation of the final binary.

2) Syntax Extension of C Language: For the format and semantics of annotations, we decided to adapt the concepts of Java Information Flow (Jif) [22], the DLM implementation for Java. We use curly braces as token for the labels. For variable declarations, these labels have to be placed in between the type indicator and the name of the variable (cf. Listing 1). Compared to the reference implementation of Jif, in Cif we additionally had to deal with pointers of the C language. We annotate and handle pointers the same way as usual variables, i.e. when using a pointer to reference to an array element or other values, the labels of pointer and target variable have to match accordingly to DLM. However, pointer overflows reasoned by pointer calculations are not further monitored by Cif. We expect that such coding errors can be covered by additional tools, such as Astrée [7]. This tool is already used successfully for checking code of avionic equipment.

In addition to the new label tokens, we extended the syntax of the C language with five further tokens:

principal *p1, ..., pn*: This token announces all used principals to the Cif.

actsFor(*p, q*): This token statically creates a trust relation that principal *p* is allowed to act for principal *q* in the entire source code.

declassify(*variable*, {*label*}): This token allows to loosen a confidentiality policy in order to relabel variables if required. Cif checks whether the new confidentiality policy is less restrictive than the present one.

endorse(*variable*, {*label*}): This token allows to loosen an integrity policy in order to relabel variables if required. Cif checks whether the new integrity policy is less restrictive than the present one.

PC_bypass({*label*}): This token allows to relabel the PC label without further checks of correct usage.

3) Function Declaration: In the C language functions can have a separate declaration called prototype. For the declaration of functions and prototypes, we also adapted the already developed concepts from Jif. In Jif a method (the representation for a function in object-oriented languages) has four labels:

1) **Begin Label** defines the side effects of the function like access to global variables. The begin label is the initial PC label for the function's body. From a function caller's perspective the current caller's PC label needs to be *no more restrictive* than the begin label of the called function.
2) **Parameter Labels** define for each parameter the corresponding label. From a caller's perspective these parameter labels have to match with the assigned values.
3) **Return Label** defines the label of the return value of the function. In Cif a function that returns *void* cannot have a return label. From a caller's perspective the variable that receives the returned value needs to be at least equally restrictive as the return label.
4) **End Label** defines a label for the caller's observation how the function terminates. Since C does not throw exceptions and functions return equally everytime, we omitted verifications of end labels in our Cif implementation.

 ISBN: 978-1-61208-438-1

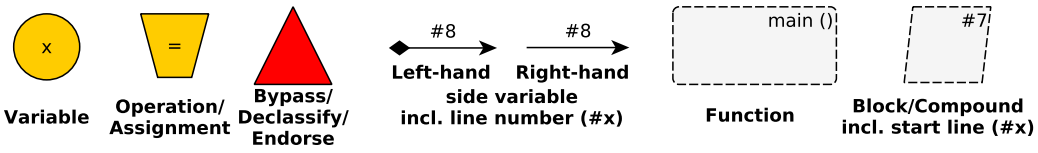


Figure 1. Legend for Flow Graphs

```
int {Alice->Bob} func {param} (int {Alice->*} param) : {Alice->*};
```
(return label: {Alice->Bob}; polymorph begin label: {param}; parameter label: {Alice->*}; end label: {Alice->*})

Listing 4. Definition of a function with DLM annotations in Cif.

Listing 4 shows the syntax for defining a function prototype with label annotations in Cif.

The definition of function labels regarding their optional prototype labels needs to be at least as restrictive, i.e. Cif allows functions to be more restrictive than their prototypes. All labels are optional augmentation to the C syntax. If the developer does not insert a label, Cif will use meaningful default labels that basically define the missing label most restrictively. Additionally, we implemented *pseudo-polymorphism*, i.e. it is possible to inherit the real label of a caller's parameter value to the begin label, return label or other parameter labels of the function. This feature is useful for the annotation of system library functions, such as *memcpy(...)* that are used by callers with divergent parameter labels and can have side effects on global variables. At this stage Cif does not support full polymorphism, i.e. the inheritance of parameter labels to variable declarations inside the function's body.

V. Use Cases

This section demonstrates the power of Cif by showing some real-world code snippets. For all examples Cif verifies the information flow modeled with the code annotations. If the information flow is valid according to the defined policy, Cif will output an unlabeled version of the C source code and a graphical representation of the flows in the source code. The format of this graphical representation is "graphml", hence, capable to further parsing and easy to import into other tools as well as documentation. Figure 1 shows the used symbols and their interpretations in these graphs. In general, the # symbol shows the line of the command's or flow's implementation in the source code.

A. Direct Assignment

The first use case presented in Listing 5 is a sequence of normal assignments.

```
1  principal Alice, Bob, Chuck;
2
3  void main {_->_;*<-*} ()
4  {
5    int {Alice->Bob, Chuck} x = 0;
6    int {Alice->Bob} y;
7    int {Alice->*} z;
8
9    y = x;
10   z = y;
11   z = x;
12 }
```

Listing 5. Sequence of Valid Direct Flows

In this example x is the least restrictive variable, y the second most restrictive variable and z the most restrictive variable. Thus, flows from $x \rightarrow y$, $y \rightarrow z$ and $x \rightarrow z$ are valid. Cif verifies this source code successfully and create the graphical flow representation depicted in Figure 2.

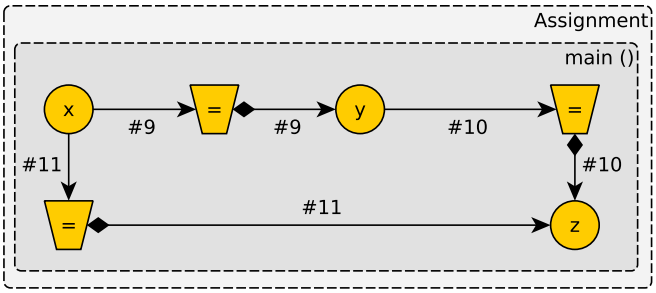


Figure 2. Flow Graph for Listing 5

B. Indirect Assignment

Listing 6 shows an example of invalid indirect information flow. Cif reports an information flow violation, since all flows in the compound environment of the true if statement need to be at least as restrictive as the label of the decision variable z. However, x and y are less restrictive and, hence, a flow to x in the assignment is not allow. Additionally, this example shows how Cif can detect coding mistakes. It is obvious that the programmer wants to prove that y is not equal to 0 to avoid the Divide-by-Zero fault. However, the programmer puts the wrong variable in the if statement. Listing 7 corrects this coding mistake. For this source code, Cif verifies that the information flow is correct. Additionally, it generates the graphical output shown in Figure 3.

```
1  principal Alice, Bob;
2
3  void main {_->_;*<-*} ()
4  {
5    int {Alice->Bob} x, y;
6    int {Alice->*} z = 0;
7
8    if(z != 0) {
9      x = x / y;
10   }
11   z = x;
12 }
```

Listing 6. Invalid Indirect Flow

```
1  principal Alice, Bob;
2
3  void main {_->_;*<-*} ()
4  {
5    int {Alice->Bob} x, y;
6    int {Alice->*} z = 0;
7
8    if(y != 0) {
9      x = x / y;
10   }
11   z = x;
12 }
```

Listing 7. Valid Indirect Flow

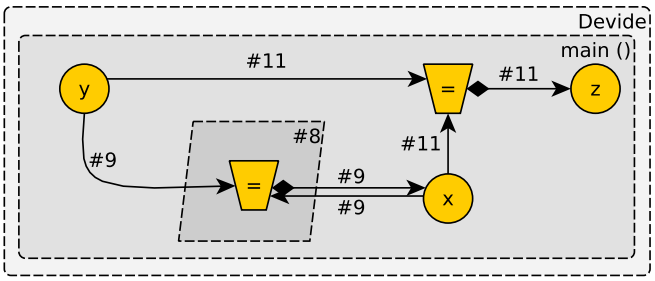


Figure 3. Flow Graph for Listing 7

Remarkable in Figure 3 is the assigment operation in line 9, represented inside the block environment of the if statement but depending on variables outside the block. Hence, Cif parses the code correctly. Also note, that in the graphical representation z depends on input of x and y, even if the source code only assigns x to z in line 11. This relation is also depicted correctly, due to the operation in line 9, on which y influences x and, thus, also z indirectly.

Another valid indirect flow is shown in Listing 8. Interesting on this example is the proper representation of the graphical output in Figure 4. This output visualizes the influence of z on the operation in the positive if environment, even if z is not directly involved in the operation.

```
1  principal Alice, Bob;
2
3  void main {_->_;*<-*} ()
4  {
5    int {Alice->Bob} x, y, z;
6
7    if(z != 0) {
8      x = x + y;
9    }
10 }
```

Listing 8. Valid Indirect Flow

C. Function Calls

A more sophisticated example is the execution of functions. Listing 9 shows a common function call using pseudo-polymorph DLM annotations. The function is called two times with different parameters on line 14 and line 15. The graphical representation of this flow in Figure 5 identifies the two independent function calls by the different lines of the code in which the function and operation is placed.

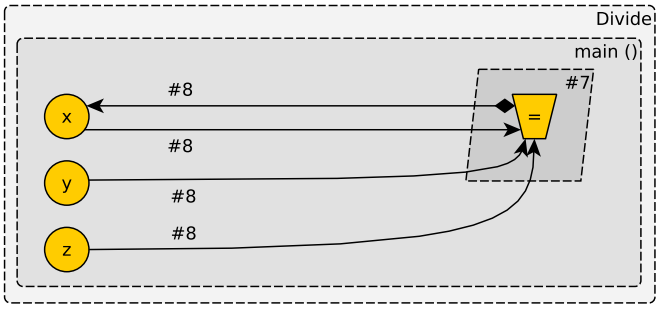


Figure 4. Flow Graph for Listing 8

```
1  principal Alice, Bob;
2
3  float {a} func (int {Alice->Bob} a,
       float {a} b)
4  {
5    return a + b;
6  }
7
8  int {*->*} main {_->_} ()
9  {
10   int {Alice->Bob} y;
11   float {Alice->Bob} x;
12   float {Alice->*} z;
13
14   x = func(y,x);
15   z = func(y,0);
16
17   return 0;
18 }
```

Listing 9. Valid Function Calls

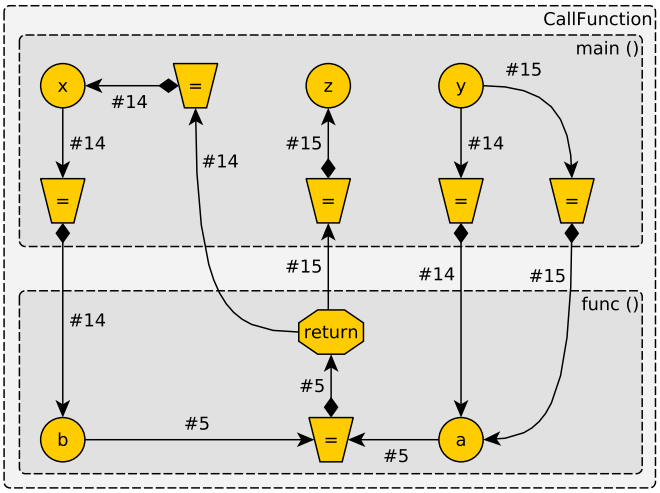


Figure 5. Flow Graph for Listing 9

D. Declassify, Endorse and Bypassing the PC

1) Using Declassify and Endorse: Stricly using DLM forces the developer to model information flows from a low restrictive source to more restrictive destinations. This unavoidably runs into the situation that information will be stored in the most restrictive variable and is not allowed to flow to some lower restricted destinations. Hence, sometimes developers need to manually declassify (for confidentiality) or endorse

 ISBN: 978-1-61208-438-1

(for integrity) variables in order to make them usable for some other parts of the program. These intended breaches in the information flow policy need special care in code reviews and, hence, it is desirable that our Cif allows the identification of such sections in an analyzable way. Listing 10 provides an example using both, the endorse and declassify statement. To allow an assignment of a to b in line 9 an endorsement of the information stored in a is necessary. The destination of this flow b is less restrictive in its integrity policy than a, since `Alice` believes that `Bob` is not allowed to modify b anymore. In line 10, we perform a similar operation with the confidentiality policy. The destination c is less restrictive than b, since `Alice` believes for b that `Bob` cannot read the information, while `Bob` can read c.

The graphical output in Figure 6 depicts both statements correctly, and marks them with a special shape and color in order to attract attention to these elements.

```
1  principal Alice, Bob;
2
3  void main {_->_;*<-*} ()
4  {
5    int {Alice->*; Alice<-Bob} a;
6    int {Alice->*; Alice<-*}   b;
7    int {Alice->Bob; Alice<-*} c;
8
9    b = endorse(a, {Alice->*;
                     Alice<-*});
10   c = declassify(b, {Alice->Bob;
                     Alice<-*});
11 }
```

Listing 10. Endorse and Declassify

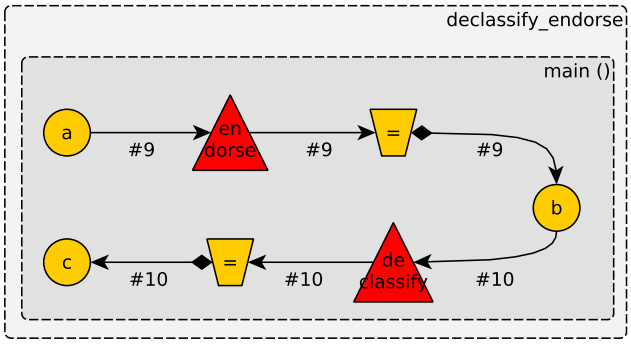


Figure 6. Flow Graph for Listing 10

2) Bypassing the PC label: In example of Listing 11 we use a simple login function to prove a user-provided uID and $pass$ against the stored login credentials. If the userID and the password match, a global variable $loggedIn$ shall be set to 1 to identify other parts of the application that the user is logged in. This status variable is owned by the principal $System$ and only this principal is allowed to read the variable. The input variables uID and $pass$ are both owned by the principal $User$. The interesting lines of this example are lines 16–18, i.e. the conditional block that checks whether the provided credentials are correct and change the status variable $loggedIn$. Note, that this examples also presents Cif's treatment of pointers on the `strcmp` function. Due to the variables in the if statement, the PC label inside the following block is `System-> & User->`. However, this PC is not more restrictive than the label of $loggedIn$ labeled with `System->`. Hence, Cif would report an invalid indirect information flow on this line. To finally allow this actual violation of the information flow, the programmer needs to manually downgrade or bypass the PC label as shown in line 17. In order to identify such manual modifications of the information flow policy, Cif also adds this information in the generated graphical representation by using a red triangle indicating the warning (see Figure 7). This shall enable code reviewers to identify the critical sections of the code to perform their (manual) review on these sections intensively.

```
1  principal User, System;
2
3  int {System->*} loggedIn = 0;
4
5  int {*->*} strcmp {*->*}
       (const char {*->*} *str1,
        const char {*->*} *str2)
6  {
7    for(; *str1==*str2 && *str1; str1++,
         str2++) ;
8    return *str1 - *str2;
9  }
10
11 void checkUser {System->*}
       (const int {User->*} uID,
        const char {User->*} * const pass)
12 {
13   const int {System->*} regUID = 1;
14   const char {System->*} const
         regPass[] = "";
15
16   if(regUID == uID &&
         !strcmp(regPass, pass)) {
17     PC_bypass({System->*});
18     loggedIn = 1;
19   }
20 }
```

Listing 11. Login Function

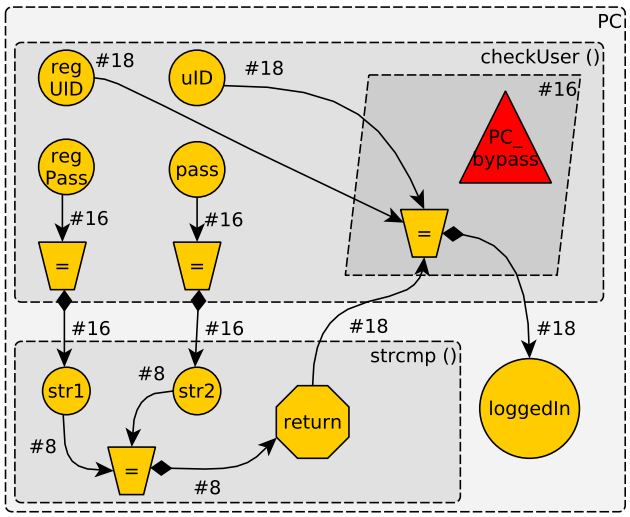


Figure 7. Flow Graph for Listing 11

VI. Conclusion

In this paper we presented Cif, a static verification tool to check information flows modeled directly in C source code. Cif is an implementation of the Decentralized Label Model (DLM) [9] for the programming language C. To the best of our knowledge this is the first time DLM is applied to the C language. During the application of DLM to C, we tried to stick to the reference implementation of Java/Jif. However, we had to discuss and solve some language-specific issues, such as pointer arithmetic or the absence of exceptions. Additionally, we added the possibility of defining annotations to function prototypes only, in case a library's source code is not available for public access. We then also introduced rules for differing annotations of function prototypes compared to function implementations.

In various code snippets, we discussed information flows as they appear commonly in C implementations. Cif is able to verify all of these examples successfully. In case of valid information flows through the entire source code, Cif generates a graphical representation of the occuring flows and dependencies. This covers direct assignments of variables, logical and arithmetic operations, indirect dependencies due to decision branches and function calls. DLM also introduces operations to intentionally loosen the strict flow provided by the model. These methods are called endorsement and declassification. Cif also implements these possibilities and specially marks them inside the graphical representation. Since DLM-annotated source code shall reduce the efforts of manual code reviews, these graphical indications allow to identify critical parts of the source code. Such parts usually require then special investigation during code reviews.

We also used Cif to implement and verify a larger internal demonstrator project. For high assurance and verification reasons, the demonstrator uses a loosely coupled software design (inspired by [5]) composed of several components with local, analyzable security services working together to provide the software's services. The information flow modeling using annotations helped us to concentrate the implementation on the component's functional purposes only. Furthermore, the information flow evaluation of the component identified several issues in the source code and, finally, could increase the code quality significantly. Particularly, the visualization of indirect flows, e.g., Listing 7 or Listing 8, and function calls, e.g., Listing 9, was very useful during the evaluation. Additionally, this activity showed that Cif is able to cover larger projects, too.

Finally, Cif allows to verify information flows in application implementations with a high level of assurance. This pioneers to create sufficient evidence for security evaluation on high assurance levels, e.g. EAL 7 of the Common Criteria.

Acknowledgment

This work was supported by the ARTEMiS Project SeSaMo, the European Union's 7^{th} Framework Programme project EURO-MILS (ID: ICT-318353), the German BMBF project SiBASE (ID: 01IS13020) and the project EMC2 (grant agreement No 621429, Austrian Research Promotion Agency (FFG) project No 84256,8 and German BMBF project ID 01IS14002). We want to express our gratitude to our project partner at the Danish Technical University, in particular Flemming Nielson. We also thank Kawthar Balti for her input.

References

[1] EUROCAE/RTCA, "ED-12C/DO-178C: Software Considerations in Airborne Systems and Equipment Certification," European Organisation for Civil Aviation Equipment / Radio Technical Commission for Aeronautics, Tech. Rep., 2012.

[2] ——, "ED-80/DO-254, Design Assurance Guidance for Airborne Electronic Hardware," European Organisation for Civil Aviation Equipment / Radio Technical Commission for Aeronautics, Tech. Rep., 2000.

[3] H. Butz, "The Airbus Approach to Open Integrated Modular Avionics (IMA): Technology, Methods, Processes and Future Road Map," Hamburg, Germany, March 2007.

[4] J. Rushby, "Separation and Integration in MILS (The MILS Constitution)," SRI International, Tech. Rep. SRI-CSL-08-XX, Feb. 2008.

[5] K. Müller, M. Paulitsch, S. Tverdyshev, and H. Blasum, "MILS-Related Information Flow Control in the Avionic Domain: A View on Security-Enhancing Software Architectures," in *Proc. of the* 42^{nd} *International Conference on Dependable Systems and Networks Workshops (DSN-W)*. Boston, MA, USA: IEEE, Jun. 2012, pp. 1–6.

[6] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, and L. K. Rierson, "A Practical Tutorial on Modified Condition/Decision Coverage," NASA, Tech. Rep. May, 2001.

[7] D. Kästner, S. Wilhelm, S. Nenova, P. Cousot, R. Cousot, J. Feret, A. Miné, X. Rival, L. Mauborgne, A. Angewandte, I. Gmbh, S. Park, and D. Saarbrücken, "Astrée: Proving the Absence of Runtime Errors," in *Proc. of the Embedded Real Time Software and Systems (ERTS2'10)*, Toulouse, France, 2010, pp. 1–9.

[8] J. C. King, "Symbolic Execution and Program Testing," *Communications of the ACM*, vol. 19, no. 7, pp. 385–394, Jul. 1976.

[9] A. C. Myers and B. Liskov, "A Decentralized Model for Information Flow Control," in *Proc. of the* 16^{th} *ACM symposium on Operating systems principles (SOSP'97)*. Saint-Malo, France: ACM, 1997, pp. 129–142.

[10] A. C. Myers, "JFlow: Practical Mostly-Static Information Flow Control," in *Proc. of the* 26^{th} *ACM Symposium on Principles of Programming Languages (POPL'99)*. ACM, Jan. 1999, pp. 228–241.

[11] Common Criteria, "Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and general model," 2009.

[12] ——, "Common Criteria for Information Technology Security Evaluation - Part 2: Security functional components," 2009.

[13] ——, "Common Criteria for Information Technology Security Evaluation - Part 3: Security assurance components," 2009.

[14] EASA, "Notification of a Proposal to Issue a Certification Memorandum: Software Aspects of Certification," EASA, Tech. Rep., Feb. 2011.

[15] A. Sabelfeld and A. C. Myers, "Language-Based Information-Flow Security," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 5 – 19, Jan. 2003.

[16] H. R. Nielson, F. Nielson, and X. Li, "Disjunctive Information Flow," DTU Compute, Technical University of Denmark, Denmark, 2014.

[17] D. Greve, "Data Flow Logic: Analyzing Information Flow Properties of C Programs," in *Proc. of the* 5^{th} *Layered Assurance Workshop (LAW'11)*. Orlando, Florida, USA: Rockwell Collins, Research sponsored by Space and Naval Warfare Systems Command Contract N65236-08-D-6805, Dec. 2011.

[18] D. Greve and S. Vanderleest, "Data Flow Analysis of a Xen-based Separation Kernel," in *Proc. of the* 7^{th} *Layered Assurance Workshop (LAW'13)*. New Orleans, Louisiana, USA: Rockwell Collins, Research sponsored by Space and Naval Warfare Systems Command Contract N66001-12-C-5221, Dec. 2013.

[19] U. P. Khedker, A. Sanyal, and B. Karkare, *Data Flow Analysis: Theory and Practice*. CRC Press, 2009.

[20] S. Chong and A. C. Myers, "Decentralized Robustness," in *Proc. of the* 19^{th} *IEEE Computer Security Foundations Workshop (CSFW'06)*. Washington, D.C, USA: IEEE, Jul. 2006, pp. 242–253.

[21] L. Zheng and A. C. Myers, "End-to-End Availability Policies and Noninterference," in *Proc. of the* 18^{th} *IEEE Computer Security Foundations Workshop (CSFW'05)*. IEEE, Jun. 2005, pp. 272–286.

[22] S. Chong, A. C. Myers, K. Vikram, and L. Zheng, *Jif Reference Manual*, http://www.cs.cornell.edu/jif/doc/jif-3.3.0/manual.html, Feb 2009, jif Version: 3.3.1; [retrieved: Sept, 2015].

 ISBN: 978-1-61208-438-1

Minimizing Attack Graph Data Structures

Peter Mell
National Institute of Standards and Technology
Gaithersburg, MD United States
email:peter.mell@nist.gov

Richard Harang
U.S. Army Research Laboratory
Adelphi, MD United States
email:richard.e.harang.civ@mail.mil

Abstract— **An attack graph is a data structure representing how an attacker can chain together multiple attacks to expand their influence within a network (often in an attempt to reach some set of goal states). Restricting attack graph size is vital for the execution of high degree polynomial analysis algorithms. However, we find that the most widely-cited and recently-used 'condition/exploit' attack graph representation has a worst-case quadratic node growth with respect to the number of hosts in the network when a linear representation will suffice. In 2002, a node linear representation in the form of a 'condition' approach was published but was not significantly used in subsequent research. In analyzing the condition approach, we find that (while node linear) it suffers from edge explosions: the creation of unnecessary complete bipartite sub-graphs. To address the weaknesses in both approaches, we provide a new hybrid 'condition/vulnerability' representation that regains linearity in the number of nodes and that removes unnecessary complete bipartite sub-graphs, mitigating the edge explosion problem. In our empirical study modeling an operational 5968-node network, our new representation had 94 % fewer nodes and 64 % fewer edges than the currently used condition/exploit approach.**

Keywords- attack graph; complexity analysis; data structures; minimization; representation; security.

I. INTRODUCTION

An attack graph is a representation of how an attacker can chain together multiple attacks to expand their influence within a network (often in an attempt to reach some set of goal states) [1]. Among other things, an attack graph can be used to calculate the threat exposure of critical assets, prioritize vulnerability remediation, optimize security investments, and as a tool to guide post-compromise forensic activities. Restricting attack graph size is vital for both human visualization of sub-graphs and the execution of high degree polynomial analysis algorithms. Early attack graph research used a 'state enumeration' representation [2] that would record all possible orderings by which an attacker could exploit a set of vulnerabilities, and hence grow at a factorial rate (exponential with some modifications). This rapid growth was mitigated in 2002 by a 'condition-oriented' approach providing a linear number of data objects with a quadratic worst-case number of relationships (with respect to the number of hosts in the original network) [3]. A major tenet of this approach was the assumption of 'monotonicity', which stated that an attacker would never lose a privilege once it was gained and any increase in privilege would not negate other gained privileges. This removed the need to account for the order in which attacks are initiated, and so reduced the complexity of the representation.

Subsequent research modified this model to make it attack- focused and enable humans to visually follow the steps within an attack more easily [2]. This hybrid 'condition/exploit' model [4] has been used extensively since 2003 for attack graph research. Unfortunately, we find that this model results in redundant data encodings, under which the worst-case graph size become quadratic in the number of nodes. Thus, over a decade of attack graph research has used a quadratic representation when a linear one was available in the literature.

However, even had the node linear condition-oriented approach been adopted, we find that it suffers from avoidable edge explosions (the creation of unnecessary complete bipartite sub-graphs) under certain conditions. These edge explosions add a quadratic factor to worst-case edge growth based on the maximum number of attacker privileges on any one host.

These size complexity issues are not always readily apparent from visual inspection of small example graphs. Example attack graphs in the literature to date have often contained a small number of dissimilar hosts with limited per-host attack surfaces and thus do not reveal the worst-case growth in both nodes and edges that we have observed. In large enterprise networks, however, where hosts have both large and diverse attack surfaces and are vulnerable to high-level compromise (thus yielding a high number of post-conditions), these complexity issues become much more evident.

To address these weaknesses, we provide a new hybrid 'condition/vulnerability' representation that regains linearity in the number of nodes and that does not suffer from the edge explosion problem. In our empirical study of a network model derived from an operational 5968-node network, our new representation had 94 % fewer nodes and 64 % fewer edges than the most widely cited recent approach in our surveyed literature (the condition/exploit model).

This reduced graph size will increase the speed of automated analysis, even making some algorithms tractable under certain scenarios. This can be true for higher polynomial complexity algorithms such as the cubic algorithms in [3] and certainly for metrics derived from attack graphs that grow either exponentially or with high polynomial degree [5]-[8] (as often occurs when graphs containing cycles must be rendered acyclic for the purposes of probabilistic modeling). The reduced size can also aid in human visualization and analysis of select sub-graphs of interest.

 ISBN: 978-1-61208-438-1

The development of the work is as follows. In Section 2, we survey past attack graph representations and describe them in terms of four major categories (while citing minor variations). For each category, we provide a description, theoretical analysis of worst-case growth, and an example graph from previously published work. Section 3 then provides our two new representations that improve upon the worst-case node and edge growth of the other representations. Section 4 provides a theoretical analysis of set of analyzed approaches. Section 5 provides empirical results using a network model based on an operational network where we compare the attack graph sizes using the different approaches. Section 6 concludes the paper.

II. SURVEY OF ATTACK GRAPH REPRESENTATIONS

Papers discussing some abstraction of the attack graph idea began appearing as early as 1996 [9]-[12]. The first widely used representation was the 'state enumeration' approach ([2] and [13]-[16]) that had the unfortunate characteristic that it could grow faster than exponential. In 2002, the 'condition-oriented' approach was published with the express purpose of reducing the graph representation size down to polynomial complexity [3]. In 2003, the 'exploit-oriented' approach was published [17] to enhance the human readability of the graphs compared to the condition-oriented approach [2]. While not discussed in the literature, we will show that the exploit-oriented approach suffers high polynomial growth rates. This may explain why, in the same year, our surveyed literature moved to a more efficient hybrid 'condition/exploit-oriented' approach [4] (which we find still has a growth rate higher than that of the condition-oriented approach). Our literature survey shows that this approach has been used extensively since 2003 and is the predominant representation (e.g., [1], [4], [7], and [18]-[20]). The following subsections discuss each of these approaches in detail. It should be noted that each representation encapsulates the same underlying knowledge but using a different abstraction. For each type of representation, we analyze the worst-case growth rate of a resulting attack graph with respect to the number of nodes and edges. We define *h* to be the number of hosts in the associated physical network, *v* to be the maximum number of vulnerabilities on any one host, and *c* to be the maximum number of attacker privileges that can be achieved on any one host from the set of available vulnerabilities. For the graph size complexity analyses, we assume that there is a unique set of pre-conditions for each attack.

A. State Enumeration

State enumeration was the first widely used attack graph approach. Its distinguishing feature is that it explicitly accounts for the different orderings in which an attacker may launch attacks. There are two types. One type uses nodes to represent attacks and edges to represent post-conditions resulting in attacker privilege [13]-[15]. The other type uses nodes to represent attacker privilege and edges to represent attacks ([2] and [16]).

The graph design contains multiple layers of nodes, regardless of the particular node and edge semantics. The initially available set of conditions or attacks are presented at the top layer. Each subsequent lower layer represents the possible decisions that an attacker could make. Directed edges connect the nodes at one layer to the available nodes at the next lower layer. There are no edges between nodes at a particular layer. An attack scenario begins at the top layer and works its way down with the node chosen in each layer representing an attacker's next decision.

1) Complexity Analysis

Assume that *v* and *c* equal 1. At the top layer (labelled 0), any of h hosts may be selected as the first node in the attack path. At layer 1, there are *h*-1 hosts that can be attacked from each of the h start nodes. For each node at layer 2 there are *h*-2 hosts that can be attacked, and so forth. This creates a rapidly expanding tree where the number of nodes and edges increases as O(*h*!), yielding a worst-case factorial growth rate for state enumeration graphs.

Most approaches attempt to prune this tree to focus only on subtrees of interest (e.g., those leading to some goal state). This can limit the growth, allow limited reuse of nodes, and can migrate the structure from a tree to a hierarchical directed graph with no loops (see Figure 1). Despite such optimizations, the growth rate of this approach is still worst-case exponential [2].

From an empirical perspective, such graphs become too large to be practical. For example, a network with just 5 hosts and 8 available exploits produced an attack graph with 5948 nodes and 68 364 edges [15].

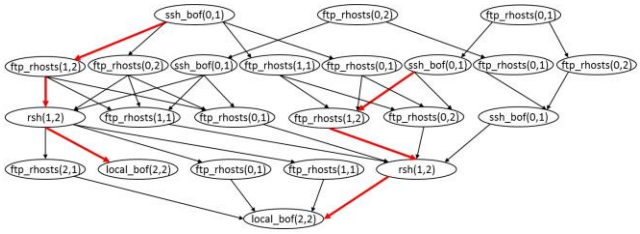


Figure 1. Example State Enumeration Attack Graph

Figure 1 shows an example pruned state enumeration attack graph, from [2], derived from an example with 2 target hosts running 2 services each, a single attacking host, and 4 unique attack types. Notice that the representation uses the previously discussed modifications to avoid factorial growth rate. The large red arrows highlight a remaining inefficiency by showing an example of path duplication in the graph (discussed in [2]).

Additional variations include [21]-[24].

B. Condition-Oriented

The condition-oriented approach was introduced in [3] as a method to reduce the representational complexity of the state enumeration approach by using the assumption of attacker privilege 'monotonicity'. This assumption implies that an attacker never loses a privilege once it is gained, and any increased privilege will not negate any other privileges. In practice, it means that (unlike in the state enumeration approach) there is no longer any need to track the order in which attacks are initiated. This assumption has been found

 ISBN: 978-1-61208-438-1

to be reasonable in most cases [2] and has been adopted by almost all subsequent attack graph representations.

The work of [2] introduced a graphical representation to the approach. Each 'condition node' represents some state of attacker privilege (e.g., execute as superuser on host x). An edge from node a to node b with label c represents that pre-condition a is necessary (but not necessarily sufficient) for attack c to provide privilege b. Sufficiency to gain privilege b is obtained if the pre-conditions of all edges into node b, labeled with c, are satisfied. By grouping the in-edges to each node by their edge labels, we can see that each group represents a possible attack, and each node is thus expressed in a variant of disjunctive normal form (DNF): in-edges within a group provide conjunctional logic and the distinct groups form logical disjunctions. Note that unlike general DNF statements, negation is not represented.

One limitation of the approach in [3] is that a single attack on a host that can be instantiated by different sets of pre-conditions must be represented by multiple attack instances named differently (using the edge names) to enable disjunctional logic. Distinct attack instances, involving different hosts, may be named similarly with no ambiguity.

1) Complexity Analysis

The condition-oriented approach achieves linearity in the number of nodes, significantly reducing the complexity compared to the state enumeration approach. An attacker may have up to c distinct condition states on each of the h hosts, and thus the number of nodes is bounded by hc. Unfortunately, as each of the hc condition nodes can be connected to all hc other nodes, and each connection between two nodes may have up to v edges to represent exploiting each available vulnerability, we obtain up to $hc \times hc \times v = h^2c^2v$ edges in the graph. Normally, h is much larger than c or v for a large network (as c and v are the maximums per node, not totals) and thus we can treat c and v as constants for the complexity analysis. Thus, the condition approach is $O(h)$ in nodes and $O(h^2)$ in edges, representing an enormous improvement over the exponential state enumeration approach. However, note the multiplicative c^2 term in the number of edges. This is the result of unnecessary complete bipartite sub-graphs forming under certain conditions. We analyze these edge explosion situations in more detail in Section 4.

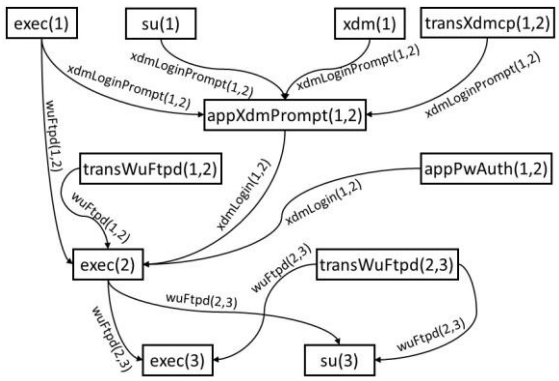


Figure 2. Example Condition-Oriented Attack Graph

Figure 2 shows an example condition-oriented attack graph, from [2], derived from a network with 2 target hosts, a single attacking host, and 3 unique attack types. Note the reduced complexity compared to Figure 1 although, as stated previously, these small example graphs are primarily intended to illustrate the methods, and do not demonstrate the differences in worst-case size complexities.

Additional variations include [21] and [25].

C. Exploit-Oriented

The exploit-oriented approach represents attacks as nodes and states of attacker privilege as edges ([2] and [17]) to ease visual analysis compared to the condition-oriented approach. Note that each 'attack node' is labeled with the host launching the attack and the host receiving the attack (which can be the same host for local attacks). This dual labeling on attack nodes will cause significant representational inefficiencies. The in-edges to a node represent the pre-conditions for launching an attack and the out-edges represent the post-conditions of the exploit. All out-edge post-conditions of a node are satisfied if and only if all the in-edge pre-conditions are satisfied.

Explicit representation of disjunction is not available and so in the presentation in the literature, attacks that can be instantiated by distinct sets of pre-conditions must be represented by multiple nodes. However, by applying a similar edge grouping approach as in the condition-oriented approach, this duplication of nodes can be avoided. Since the edges in this approach are already labeled with the post-condition names they must be additionally labeled with a grouping name (a name for the group of pre-conditions that will enable exercising the related exploit). While this modification is not discussed in the literature, we assume this optimization to represent the approach as efficiently as possible. Without this optimization, the number of nodes would increase by a factor of c.

1) Complexity Analysis

We now examine the worst-case growth rate of some arbitrary attack graph. With respect to nodes, the graph can grow as large as h^2v. Each of the h hosts can attack all h hosts with v different attacks (assuming just one attack per vulnerability) resulting in h^2v nodes. With respect to edges, the graph can grow as large as h^3v^2. Consider a single node where b represents the attack target. This node can have a connection to each node where the attack source is b. There will be hv nodes with attack source b. Each connection though can be made up of c edges. Thus, each node can create hvc out-edges. Since the number of nodes is h^2v, this leads us to $h^2v \times hvc = h^3v^2c$ edges. Treating c and v as constants compared to h, the exploit approach is $O(h^2)$ in nodes and $O(h^3)$ in edges. This is an enormous increase in graph size from the condition approach, however it still outperforms the exponential state enumeration approach.

Figure 3 shows an example exploit-oriented attack graph, from [2], derived from the same network as Figure 2. The example condition-oriented graph has 11 nodes and 12 edges while the example exploit-oriented graph as 6 nodes and 13 edges. Despite the reduction in graph size demonstrated in this example, we will empirically show that the exploit-

 ISBN: 978-1-61208-438-1

oriented graphs can grow much larger than the condition-oriented graphs in enterprise networks.

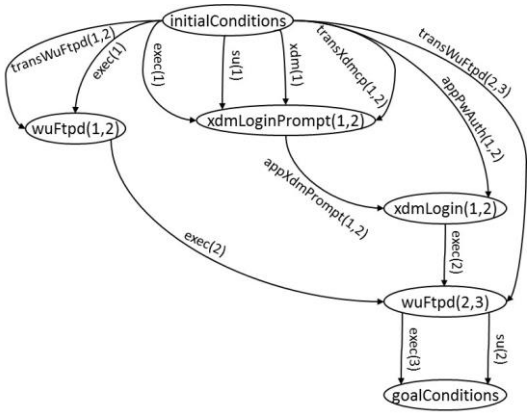


Figure 3. Example Exploit-Oriented Attach Graph

D. Hybrid Condition/Exploit-Oriented

The hybrid condition/exploit-oriented approach uses two distinct types of nodes ([1], [4], [7], and [18]-[20]) representing both attacks and the states of attacker privilege, while the edges are unlabeled. The 'attack nodes' and 'condition nodes' have the same semantics as those in the exploit-oriented graphs and the condition-oriented graphs, respectively.

This structure produces a directed bipartite graph, with attack nodes having edges to condition nodes and vice versa. However, the interpretation of the in-edges varies per type of node. Attack nodes and their post-condition out-edges are all satisfied if and only if all in-edges are satisfied (conjunction as with the exploit-oriented graphs). Condition nodes and their out-edges are satisfied if at least one in-edge is satisfied (disjunction as with multiple groups of in-edges in the condition-oriented graphs).

As in the exploit-oriented representation, the problem of a single exploit that can be instantiated with multiple sets of pre-conditions is not well addressed in the literature. In a naïve implementation, a single exploit must be divided into multiple attack node instances, one for each distinct set of pre-conditions. However, by applying the same optimization as before (again, not previously presented in the literature) where we allow condition node to attack node edges to be labeled with a group name, we can avoid this multiplication, and we assume this optimization throughout. As in the condition approach, the interpretation is disjunction among the groups and conjunction within a group. Without this optimization, the worst-case number of attack nodes would increase by a factor of c (as with the exploit approach).

1) Complexity Analysis

We now examine the worst-case growth rate of some arbitrary attack graph. With respect to nodes, the graph can grow as large as $hc+h^2v$. There will be hc condition nodes as derived in Section 2.2.1 and there will be h^2v attack nodes as derived in Section 2.3.1. With respect to edges, the condition and attack nodes form a directed bipartite graph. We first explore the set of attack to condition node edges. Each attack node has a single target host as discussed in Section 2.3. Each attack node then can at most activate c condition nodes on the target host where each activation creates an attack to condition node edge. Since there are up to h^2v attack nodes, we can then have up to $h^2v \times c = h^2vc$ attack node to condition node edges. Similarly, each condition node is mapped to a host, say a, and thus may have an edge to any of the hv exploit nodes where the attack source is a. Since there are hc condition nodes and up to hv edges per node, we get a total of $hc \times hv = h^2vc$ condition to exploit node edges. Summing the two types of edges, we get $h^2vc+h^2vc=2h^2vc$.

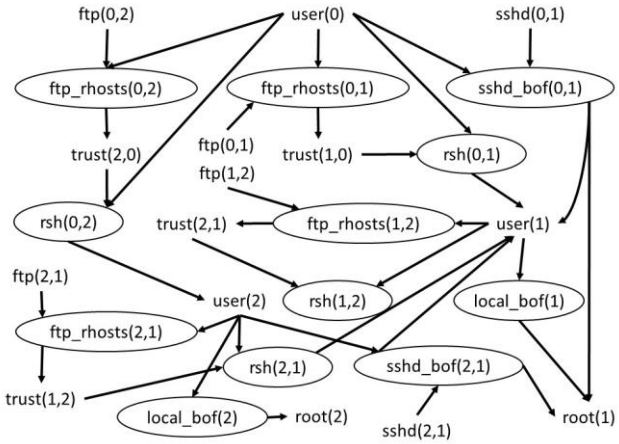


Figure 4. Example Condition/Exploit-Oriented Attack Graph

Figure 4 shows a condition/exploit-oriented attack graph from an example provided in [4]. This was derived from the same network as in Figure 1. The circled nodes are the attack nodes and the un-circled nodes are the condition nodes. Notice the relative simplicity in comparison with the state enumeration approach in Figure 1. Again though, the size of these small examples doesn't demonstrate complexity growth on large enterprise networks.

Additional variants include [26]-[28].

III. VULNERABILITY-BASED REPRESENTATIONS

Our contribution is the idea of representing the vulnerabilities on specific hosts explicitly within attack graphs. From a visualization point of view, this makes it easy to see how a chain of vulnerabilities (and hosts) can be compromised. From a graph complexity point of view, the vulnerability nodes replace the use of the attack nodes thereby lowering node complexity to linear (from quadratic). Intuitively, where exploit nodes must contain references to both the source and target hosts in an attack step, and thus grow potentially quadratically in highly connected networks, the vulnerability nodes only reference the exploitable host, and so grow only linearly. We represent attacks within the edges where they can take advantage of the fact that edges inherently have sources and targets (similar to the condition approach).

This approach leads to two new representations that build upon one another. We first describe a vulnerability-oriented approach where we replace the attack nodes from the exploit-oriented approach with vulnerability nodes. This reduces node complexity from quadratic to linear and edge

 ISBN: 978-1-61208-438-1

complexity from cubic to quadratic. We then point out how the vulnerability-oriented approach suffers from similar quadratic edge explosion scenarios as the condition approach (but this time relative to *v*, not *c*).

We then build upon the vulnerability approach to provide a hybrid condition/vulnerability representation that has a linear number of nodes, quadratic number of edges, and that does not suffer from quadratic edge explosion (in either *v* or *c*). It also, like the hybrid condition/exploit approach, improves on the human readability of the condition-oriented approach.

A. Vulnerability-Oriented

Our first approach is analogous to the exploit-oriented approach except that we replace the attack nodes with vulnerability nodes. A vulnerability node is labeled with a vulnerability name and the relevant location in the network (usually but not necessarily a hostname). Edges represent attacker privilege, just like in the exploit-oriented approach. In cases where multiple sets of pre-conditions can activate a particular vulnerability, we handle it with the edge grouping optimization we've presented previously. A set of pre-conditions that will activate a vulnerability are represented by a set of in-edges to a node that all have a common group name. Thus we represent attacks using groups of commonly named edges just like in the condition approach. Each node is thus expressed in DNF: in-edges within a group provide conjunctional logic and the distinct groups form logical disjunctions.

1) Complexity Analysis

We now examine the worst-case growth rate of some arbitrary attack graph. With respect to nodes, the graph can grow as large as *hv* because each of the *h* hosts can have *v* vulnerabilities. With respect to edges, each node can have *hv* outgoing connections to other nodes. Each connection can be made up of at most *c* edges. Thus, the total number of edges is at most $hv \times hv \times c = h^2v^2c$.

A disadvantage of this approach is the v^2 term in the number of edges. This is the result of unnecessary complete bi-partite sub-graphs forming under certain conditions. We analyze this edge explosion in more detail in Section 4.

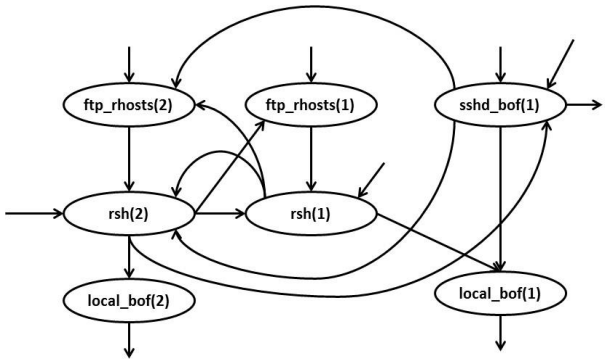


Figure 5. Example Vulnerability-Oriented Attack Graph

Figure 5 shows an example of a vulnerability-oriented graph, derived from the representation in Figure 4. For the sake of readability, edge grouping labels are omitted. Note that the number of nodes is reduced with respect to Figure 4, and all vulnerabilities are in exactly one node instead of replicated over multiple nodes with different attack sources (e.g., the attacks targeting the ftp rhosts vulnerability on node 1 in Figure 4).

B. Hybrid Condition/Vulnerability-Oriented

Our second novel approach is a hybrid approach combining condition nodes with our new vulnerability nodes. The condition nodes are analogous to those in the condition-oriented approach. The vulnerability nodes are identical to those in our vulnerability-oriented representation. We represent attacks by labeling the condition node to vulnerability node edges with the attack instances being used (including source and destination hostnames/IPs where applicable); this edge labeling is similar to that in the condition-oriented graphs. We then connect the vulnerability nodes to the condition nodes with unlabeled edges showing which post-conditions emerge as a result of exploiting the relevant vulnerability instance.

As with the hybrid condition/exploit graph, this structure produces a directed bipartite graph. For condition nodes, they and all of their attack labeled out-edges are satisfied if at least one in-edge is satisfied (disjunction). For vulnerability nodes, they and all of their unlabeled out-edges are satisfied if and only if a group of identically labeled in-edges are satisfied (conjunction within a group and disjunction between the groups). This distributed implementation of disjunctions and conjunctions enables the same DNF logic of the condition-oriented approach.

A single attack that can be instantiated with multiple sets of pre-conditions can be represented by using a different group name for each set of instantiating pre-conditions.

1) Complexity Analysis

We now examine the worst-case growth rate of an arbitrary attack graph. With respect to nodes, the graph can grow as large as $h(c+v)$. There will be *hc* condition nodes as derived in Section 2.2.1 and *hv* vulnerability nodes as derived Section 3.1.1. Thus, there are at most $hc+hv$ nodes total. With respect to edges, there can be as many as $h^2cv+hvc$. First, for the set of edges that point from vulnerability nodes to condition nodes, each of the *hv* vulnerability nodes can activate up to *c* condition nodes (since each vulnerability pertains to a host with up to *c* conditions), creating *hvc* edges. In the other direction, each of the *hc* condition nodes could allow an attack to all *hv* vulnerability nodes, producing $hc*hv$ edges. Thus, the total number of edges is $hvc+hc*hv=h^2cv+hvc$.

Treating *c* and *v* as constants compared to *h*, the condition/vulnerability approach is $O(h)$ in nodes and $O(h^2)$ in edges. This is a major improvement over the quadratic growth in the number of nodes caused by attack nodes in the exploit and condition/exploit approaches. While the complexity of the number of nodes is of the same order between the condition, vulnerability, and condition/vulnerability graphs, we note that there is no $v2$ or c^2 term in the edge growth equation for the condition/vulnerability graph. This is indicative of the fact that this approach does not suffer from the quadratic edge

 ISBN: 978-1-61208-438-1

explosion problem of the condition and the vulnerability-oriented approaches in either *v* or *c*. This is discussed in detail in Section 4.

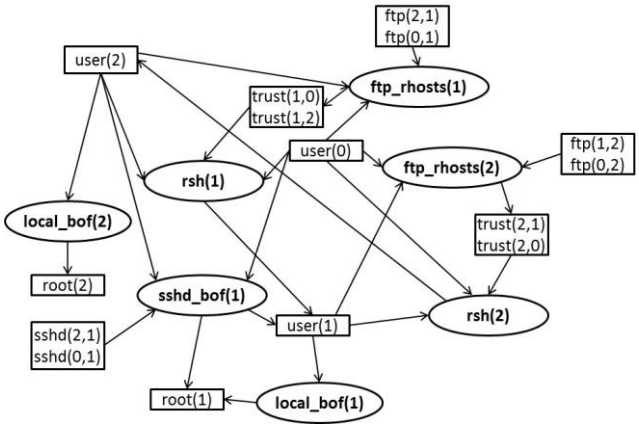


Figure 6. Example Condition/Vulnerability-Oriented Attack Graph

Figure 6 shows an example of the condition/vulnerability graph displaying the same attack graph data as in Figure 4 and Figure 5. In this example with just 3 nodes and relatively diverse attack surfaces, this data actually has a more compact representation in the vulnerability graph format. We demonstrate in Section 5 on larger scale real-world data that this advantage is not general, and in fact the condition/vulnerability graph provides significant size advantages in such cases.

IV. THEORETICAL ANALYSIS

In this section, we provide a theoretical analysis of the worst-case behavior of the representations. Note that we do not analyze the state enumeration approach given its known exponential growth rate (previously discussed). We begin with a review of the big-O complexity of each graph type and show the advantages of the condition, vulnerability, and condition/vulnerability approaches. We then follow with an analysis of the terms within the actual worst-case growth equations. These equations reveal quadratic terms in both *v* and *c* that cause the edge explosion scenarios in the condition and vulnerability approaches, respectively. We then explore in more detail when and why these avoidable edge explosion scenarios occur. We end the section with a discussion of edge explosion scenarios that are unavoidable in all of our analyzed representations (and that may reflect an inherent limit in reducing graph sizes).

A. Big-O Complexity Graph Growth Comparisons

In an attack graph, *h* is expected to grow much larger than *v* or *c* for a typical enterprise network. Note that *v* and *c* are the maximums per host (not the total number of vulnerability and conditions) and thus are usually miniscule compared to *h*. For this reason, we treat *v* and *c* as constants to derive overall complexity of each representation. These big-O measurements were derived in Sections 2 and 3 and are summarized in Table 1. The calculations showing the largest growth rates are bolded. Note, if *h* is not large relative to *v* or *c*, use the below Table 2 instead of Table 1 to determine the most efficient representation.

TABLE 1. COMPLEXITY MEASUREMENT OF ATTACK GRAPH REPRESENTATION

Representation	Nodes	Edges
Condition	$O(h)$	$O(h^2)$
Exploit	**$O(h^2)$**	**$O(h^3)$**
Vulnerability	$O(h)$	$O(h^2)$
Condition/Exploit	**$O(h^2)$**	$O(h^2)$
Condition/Vulnerability	$O(h)$	$O(h^2)$

The quadratic node growth of the exploit and condition/exploit approaches is much larger than the linear node growth of the condition and condition/vulnerability approaches. With respect to edges, the cubic edge growth of the exploit approach is larger than the quadratic growth of the other approaches. Thus, the condition, vulnerability, and condition/vulnerability approaches are the best approaches in limiting worst-case graph growth with respect to *h*.

To intuitively understand why the exploit and condition/exploit node growth is quadratic, consider that an exploit node must necessarily contain two host name labels: the attack source and the attack target. If a set of hosts A can attack a set of hosts B using a single attack, then the number of exploit nodes representing this will be |A|*|B| (i.e., quadratic growth). Contrast this to the condition/vulnerability approach where there will be only a single vulnerability node per host in B resulting in |B| vulnerability nodes (i.e., linear growth).

One optimization is to reduce graph size by consolidating groups of identical hosts (those with identical security value, vulnerabilities, and permitted connectivity) into single hosts when building the attack graph. This essentially reduces *h* and thus minimizes the graph size, but we note that it does not change the big-O complexity results nor the outcome of our comparative analyses.

B. Worst-Case Equations Graph Growth Comparisons

We now look at the actual worst-case equations that we derived in Sections 2 and 3 in order to further refine our comparison between the approaches. These equations are summarized in Table 2. The terms specifically discussed in our analysis are bolded.

TABLE 2. WORST-CASE GROWTH OF ATTACK GRAPH REPRESENTATIONS

Representation	Nodes	Edges
Condition	hc	$h^2v\mathbf{c^2}$
Exploit	h^2v	h^3v^2c
Vulnerability	hv	$h^2\mathbf{v^2}c$
Condition/Exploit	$hc+h^2v$	$2h^2vc$
Condition/Vulnerability	$\mathbf{hc+hv}$	$h^2vc+hvc$

We focus our analysis on the condition, vulnerability, and condition/vulnerability approaches since the other two approaches were shown to have larger big-O node complexities in Table 1.

With respect to node growth, the condition and vulnerability approaches will always have fewer nodes than the condition/vulnerability approach due it having the sum of

 ISBN: 978-1-61208-438-1

the former two. However, it should be noted that this increase is a small linear factor.

With respect to edge growth, however, both the condition and vulnerability graphs grow quadratically in *c* and *v*, respectively. It is these quadratic terms (not present in the condition/vulnerability edge equation) that reflect edge explosion scenarios where unnecessary complete bipartite sub-graphs are formed. We claim that they are unnecessary because the condition/vulnerability approach provides a linear representation of the same data. Visually, the condition and vulnerability approaches use complete bi-partite sub-graphs where, for the same data, the condition/vulnerability approach uses star topologies (explained in detail in the next section).

Thus overall, our theoretical analysis indicates that our condition/vulnerability approach will result in achieving the most compact attack graphs. The exploit approach was the worst (excluding the naïve state enumeration approach), suffering from cubic edge growth. The condition/exploit approach (the most commonly used in recent research in our surveyed literature) suffered from quadratic node growth in our largest term, *h*. Finally, the condition and vulnerability approaches suffered from quadratic edge explosions (in *c* and *v*, respectively) as a result of the creation of complete bi-partite sub-graphs that our condition/vulnerability approach converts to linear growth star topologies. We now explore in detail the edge explosion scenarios.

C. Avoidable Edge Explosion Scenarios

We define an edge explosion as the creation of a complete bipartite sub-graph within an attack graph due to some specific scenario. Some scenarios cause edge explosions regardless of which of our analyzed attack graph representations is used. We call these scenarios 'unavoidable' (with respect to our set of representations) and thus they are not useful for a comparative analysis. We discuss such unavoidable scenarios in the next section.

This section focuses on avoidable scenarios that create quadratic edge explosions in the condition and vulnerability representations, which are converted to linear star representations in the condition/vulnerability approach. The condition/exploit approach has similar representational advantages with respect to edge explosion scenarios. However, we do not specifically analyze this for the condition/exploit approach due to its worse quadratic node complexity but we do note that it is the dual node type representations that enable the reduction (i.e., the 'hybrid' node design).

Avoidable edge explosion can occur in both the condition and vulnerability graphs as shown by their worst-case quadratic growth in *c* and *v*, respectively. These upper bounds are approached in condition graphs whenever a single attack has multiple pre-conditions and multiple post-conditions. This also happens in vulnerability graphs when some set of vulnerabilities on a host allows subsequent exploitation of some other set of vulnerabilities (on the same or other hosts).

Both cases can be viewed in terms of directed hyperedges, representing the multi-way relationships. For example, an attack with multiple pre-conditions and post-conditions can be represented by a single directed hyperedge with the pre-conditions at the tail of the edge and the post-conditions at the head. In standard directed graphs, representing this requires the creation of a complete directed bipartite sub-graph with the pre-conditions in the edge 'tail' set and the post-conditions in the 'head' set. The representation of these hyperedges by the corresponding complete bipartite graphs is then responsible for the quadratic explosion in the number of edges. Similarly, given a set of vulnerabilities on a host where each one enables an attacker to exploit some other common set of vulnerabilities can be represented by a single directed hyperedge. In a vulnerability graph, this hyperedge would require a directed complete bipartite graph with the enabling vulnerabilities in the edge 'tail' set and the newly available vulnerability nodes in the edge 'head' set.

Hybrid node graphs, such as the condition/vulnerability and the condition/exploit graphs, represent these directed hyperedges more efficiently (i.e., linearly). To see this, consider that in the condition/vulnerability graph, each vulnerability node may represent a directed hyperedge that links multiple pre-conditions to multiple post-conditions. Each condition node may also represent a directed hyperedge that links multiple vulnerabilities on a single host to a set of common target vulnerabilities. This representational approach of a directed hyperedge forms a star graph for each hyperedge. It is then easy to see that star graphs grow linearly in the number of edges while the complete bipartite sub-graphs grow quadratically, thus enabling the size complexity advantage of the hybrid node approaches.

For the purposes of illustration, consider Figure 7 below. Conditions C_1 to C_4 form the tail of a hyperedge corresponding to a vulnerability V_a, while conditions C_5 to C_8 for the head. The resulting condition graph is complete bipartite, as each of C_1 to C_4 must be linked to each of C_5 to C_6 (Figure 7, left); by contrast, using a separate class of node to represent the vulnerability-related hyperedge in the condition/vulnerability approach allows for a much more compact representation in the form of a star graph (Figure 7, right).

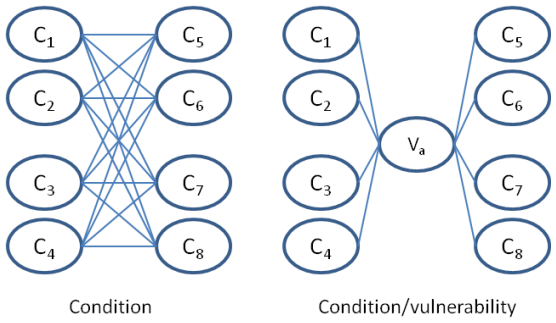


Figure 7. Unnecessary complete bipartite structures in the condition-oriented graph

Vulnerability graphs have a directly analogous representation, where a condition node may represent a hyperedge, as shown in Figure 8.

 ISBN: 978-1-61208-438-1

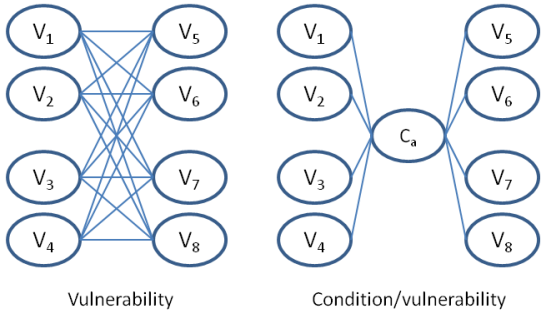


Figure 8. Unnecessary complete bipartite structures in the vulnerability-oriented approach

In the vulnerability graph, each vulnerability (V_1 to V_4) on a single host must have an edge to each vulnerability that is now accessible for attack (V_5 to V_8). In the condition/vulnerability graph, a single condition node C_a represents the attacker privileges gained by exploiting V_1 to V_4. Condition node C_a then allows exploitation of V_5 to V_8. The addition of C_a creates a linear growth star graph in place of the quadratic complete bipartite graph in the vulnerability representation.

D. *Unavoidable Edge Explosion Scenarios*

There are cases where the condition/vulnerability graph will still contain complete bipartite components and it is direct to see that the related condition or vulnerability graph will also exhibit such a component. Thus, such scenarios are unavoidable (with our set of analyzed representations). While we can't prove nonexistence of a linear representation here, we believe it unlikely and that we are pushing against inviolable data representational boundaries in trying to further reduce the size of the attack graph.

Consider a scenario where multiple distinct hyperedges have identical head or tail sets in either the condition- or vulnerability- oriented approach. This will naturally result in complete bipartite components in the condition/vulnerability representation; however, it is straightforward to see that such cases also produce complete bipartite graphs in the condition and vulnerability representations as well.

See, for example, the condition/vulnerability graphs in Figure 9. The leftmost panel depicts a situation in which two distinct vulnerabilities have identical head and tail sets (such as two identical vulnerabilities on two different hosts that each enable a common set of vulnerabilities on a third); the center depicts a situation in which the head sets are distinct but the tail sets are identical (perhaps granting host-specific post-conditions), and the rightmost pane depicts identical head sets with distinct tail sets (such as would be expected from host-specific pre-conditions with global post-conditions). In each case, it is straightforward to see that a path exists from each pre-condition to each post-condition, and so the resulting condition-oriented graph will be a complete bi-partite graph.

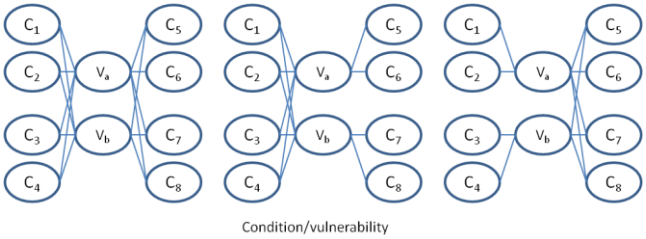


Figure 9. Condition/Vulnerability Graph Scenarios

The situation is functionally identical for vulnerability-oriented graphs. Similar complete bipartite sub-graphs within a condition/vulnerability graph will result in a completely connected bipartite sub-graphs in the related vulnerability graph.

Looking at the underlying equations, note that in the condition/vulnerability graph the number of edges is bounded as the product of *c* and *hv*, rather than being quadratic in *c*, thus requiring both *v* and *c* to grow simultaneously for a comparable edge explosion. Note that this doesn't reflect a unique weakness for the condition/vulnerability approach as both the condition and vulnerability approaches also contain *h*, *c*, and *v* in their edge equations (but there with a quadratic *c* or *v*). This worst-case scenario is only realized in the leftmost panel of Figure 9, while all three panels result in complete bipartite sub-graphs in the case of the condition-oriented graph.

V. EMPIRICAL RESULTS

We now provide an example to illustrate the performance between the different approaches using a network model. Our network model (derived in part from data from an operational network) has 5968 hosts and 7825 vulnerabilities. The vulnerabilities consist of 41 distinct types mapped to two different severity levels. We mapped 7791 vulnerability instances to confidentiality breaches and 34 instances to providing user level access.

With respect to attack post-conditions, a vulnerability was modeled as producing two post-conditions: the severity level mapped to the host name and a designator indicating that the host had some specific vulnerability exploited. This models the situation where a single attack can produce multiple post-conditions.

With respect to connectivity, we modeled all nodes as being logically connected to each other. For a start node in the attack graph, we designated one of the hosts as hostile (using one with no vulnerabilities) to represent an insider threat situation.

Table 3 provides the empirical results given the above stated scenario. To derive these results, we created an attack graph simulator using Python 2.7.6 that calculates the graph sizes using all of our analyzed representations. Note that these results are not based on the equations from Table 2 as those equations represent worst-case attack graph sizes. Here we analyze the actual sizes given the network model described above.

 ISBN: 978-1-61208-438-1

TABLE 3. EMPIRICAL RESULTS

Graph Type	Nodes	Edges
Condition	5140	436 290
Exploit	218 146	7 189 929
Vulnerability	7825	272 920
Condition/Exploit	223 285	654 435
Condition/Vuln.	12 964	233 795

As expected from the theoretical analysis, the number of nodes for the exploit and condition/exploit representations was much larger than the other approaches due to the $O(h^2)$ growth rate. The number of edges in the condition graph is almost twice that of the condition/vulnerability graph, attributable to the $O(c^2)$ growth rate of the edges. Thus based on these empirical results, the vulnerability and condition/vulnerability approaches appear the best for our scenario and are comparable (with the vulnerability approach having fewer nodes and the condition/vulnerability approach having fewer edges).

Note how in this example our condition/vulnerability approach had 94 % fewer nodes and 64 % fewer edges than the widely cited and commonly used condition/exploit approach. This illustrates how an adjustment in representation can have dramatic results in graph size.

However, if we model each attack as producing exactly one post condition, then the advantages of the condition/vulnerability approach disappear relative to the condition graph (see Table 4).

TABLE 4. SINGLE POST CONDITION EMPIRICAL RESULTS

Graph Type	Nodes	Edges
Condition	2584	218145
Exploit	218146	7189929
Vulnerability	7825	272920
Condition/Exploit	220728	436290
Condition/Vuln.	10408	225970

Here the condition graph has an advantage on the number of nodes while roughly matching the number of edges of the conditional/vulnerability approach. Thus, use of the vulnerability/condition approach does not guarantee a smaller graph than the condition representation. However, it guarantees a linear growth rate with respect to c, allowing for tighter representations given arbitrary scenarios.

Note that the vulnerability approach statistics stay the same in both Table 3 and Table 4. This is because the removed post conditions were not ones that enabled an attack to be launched (we just removed the flag that a host had a specific vulnerability exploited).

The widely-cited and used condition/exploit model was much larger in all of our scenarios because it suffers from both the $O(h^2)$ growth rate in the nodes (this is true also of the exploit approach). Had we modeled a network where the logical connectivity of the hosts was much more restricted, the node disadvantage of the condition/exploit approach would have been minimized. However, many operational networks (including this one) have large numbers of hosts with significant logical connectivity (e.g., approaching complete sub-graphs).

VI. CONCLUSIONS

For the last decade, the condition/exploit-oriented approach was the most commonly used representation in our literature survey. However, we found it to have node growth quadratic in the number of hosts on the network. This will slow down analysis algorithms that have a high polynomial degree while making visualization for humans more difficult (simply from the increased size). Interestingly, we found the previously published condition approach provided a much more compact linear node representation, but it wasn't widely adopted. This may have been because it was confusing to visually analyze since attacks are represented by collections of edges. We also discovered that it suffers from quadratic edge explosions based on the number of possible attacker privileges on a host.

To address these problems, we proposed using a vulnerability-based approach for nodes in attack graphs. This eliminates the inefficiency of the attack nodes (taking us from a quadratic to a linear node representation) while it makes the graph more intuitive to read compared to the condition approach (since any attack results in compromising a single vulnerability node as opposed to activating multiple condition nodes). Surprisingly, we found this approach to also contain an edge explosion problem but this time relative to the number of vulnerabilities on a host.

We thus developed the hybrid condition/vulnerability approach with the following advantages: linear node growth, elimination of avoidable edge explosion issues, and an easy to understand representation (due to the use of the vulnerability nodes). For arbitrary graphs, our condition/vulnerability approach provides better size guarantees with respect to edge growth while only having a small linear penalty on node growth.

Despite this, the condition and vulnerability approaches are still viable representation options (linear in node growth and quadratic in edge growth). Even with the quadratic edge explosion possibilities, they can be used when it is known that a particular scenario will not suffer significantly from this problem. Perhaps the best argument for using these two approaches is simply that they have a single interpretation for the nodes. This should facilitate the application of standard graph algorithms for analysis, something not available with the currently used hybrid condition/exploit approach or our hybrid condition/vulnerability approach. Given the simple interpretation of our vulnerability approach, it is a candidate for exploration in this area, which may be addressed in future work.

Lastly and most importantly, we emphasize that the research community should move away from using attack nodes (as found in both the exploit representation and the hybrid condition/exploit representation) since the attack nodes add a quadratic factor to the worst-case node growth equations. Moving to a much more compact node linear representation (regardless of the specific choice) may catalyze the research community by opening the door to

 ISBN: 978-1-61208-438-1

previously intractable algorithmic analyses and facilitating human analysis of specific features.

REFERENCES

[1] A. Singhal and X. Ou, "Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs," National Institute of Standards and Technology Interagency Report 7788, 2011.

[2] S. Noel and S. Jajodia, "Managing Attack Graph Complexity Through Visual Hierarchical Aggregation," in Workshop on Visualization and Data Mining for Computer Security, Fairfax, 2004, pp. 109-118.

[3] P. Ammann, D. Wijesekera and S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis," in ACM Conference on Computer and Communications Security, Washington, D.C., 2002, pp. 217-224.

[4] S. Noel, S. Jajodia, B. O'Berry and M. Jacobs, "Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs," in Computer Security Applications Conference, Las Vegas, 2003, pp. 86-95.

[5] M. Frigault, L. Wang, A. Singhal and S. Jajodia, "Measuring Network Security Using Dynamic Bayesian Network," in Proceedings of the 4th ACM Workshop on Quality of Protection, 2008, pp. 23-30.

[6] L. Wang, T. Islam, T. Long, A. Singhal and S. Jajodia., "An Attack Graph-Based Probabalistic Security Metric," in Data and Applications Security XXII, Springer, 2008, pp. 283-296.

[7] N. Idika and B. Bhargava, "Extending Attack Graph-Based Security Metrics and Aggregating Their Application," IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 1, 2012, pp. 75-85.

[8] J. Homer, X. Ou and D. Schmidt, "A Sound and Practical Approach to Quantifying Security Risk in Enterprise Networks," Kansas State University Technical Report, 2009.

[9] M. Dacier, Y. Deswarte and M. Kaaniche, "Quantitative Assessment of Operational Security: Models and Tools," LAAS Research Report 96493, 1996.

[10] I. Moskowitz and M. Kang, "An Insecurity Flow Model," in New Security Paradigms Workshop, 1997, pp. 61-74.

[11] C. Meadows, "A Respresentation of Protocol Attacks for Risk Assessment," Network Threats, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 38, 1998, pp. 1-10.

[12] C. Phillips and L. Swiler, "A Graph-Based System for Network-Vulnerability Analysis," in Proceedings of the 1998 Workshop on New Security Paradigms, Charlottesville, 1998, pp. 71-79.

[13] R. Ortalo, Y. Deswarte and M. Kaaniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security," IEEE Transactions on Software Engineering, vol. 25, no. 5, 1999, pp. 633-650.

[14] L. Swiler, C. Phillips, D. Ellis and S. Chakerian, "Computer-Attack Graph Generation Tool," in DARPA Information Survivability Conference, Anaheim, 2001, pp. 307-321.

[15] O. Sheyner, J. Haines, S. Jha, R. Lippman and J. Wing, "Automated Generation and Analysis of Attack Graphs," in IEEE Symposium on Security and Privacy, Washington D.C., 2002, pp. 273-284.

[16] S. Jha, O. Sheyner and J. Wing, "Two Formal Analyses of Attack Graphs," in IEEE Computer Security Foundations Workshop, Cape Breton, 2002, pp. 49-63.

[17] S. Jajodia, S. Noel and B. O'Berry, "Topological Analysis of Network Attack Vulnerability," in Managing Cyber Threats: Issues, Approaches and Challenges, Kluwer Academic Publisher, 2003, pp. 247-266.

[18] S. Noel and S. Jajodia, "Measuring Security Risk of Networks Using Attack Graphs," International Journal of Next-Generation Computing, vol. 1, no. 1, 2010, pp. 135-147.

[19] J. Pamula, P. Ammann, S. Jajodia and V. Swarup, "A Weakest-Adversary Security Metric for Network Configuration Security Analysis," in Workshop on Quality of Protection, Alexandria, 2006, pp. 31-38.

[20] L. Wang, S. Noel and S. Jajodia, "Minimum-Cost Network Hardening Using Attack Graphs," Computer Communications, 2006, pp. 3812-3824.

[21] B. Schneier, "Attack trees," Dr. Dobb's journal, 1999, pp. 21-29.

[22] B. Kordy, S. Mauw, S. Radomirović and P. Schweitzer, "Foundations of attack–defense trees," in Formal Aspects of Security and Trust, Springer , 2011, pp. 80-95.

[23] V. Gorodetski and I. Kotenko, "Attacks against computer network: Formal grammar-based framework and simulation tool," in Recent Advances in Intrusion Detection, 2002, pp. 219-238.

[24] R. W. Ritchey and P. Ammann, "Using model checking to analyze network vulnerabilities," in 2000 IEEE Symposium on Security and Privacy, 2000, pp. 156-165.

[25] J. Dawkins and J. Hale, "A systematic approach to multi-stage network attack analysis," in Proceedings, Second IEEE International Information Assurance Workshop, 2004, pp. 48-56.

[26] N. Poolsappasit, R. Dewri and I. Ray, "Dynamic security risk management using bayesian attack graphs," IEEE Transactions on Dependable and Secure Computing, 2012, pp. 61-74.

[27] D. Koller and N. Friedman, Probabilistic graphical models: principles and techniques, MIT Press, 2009.

[28] S. J. Templeton and K. Levitt, "A requires/provides model for computer attacks," in Proceedings of the 2000 Workshop on New Security Paradigms, 2001, pp. 31-38.

[29] R. Lippmann, K. Ingols, C. Scott and K. Piwowarski, "Validating and Restoring Defense in Depth Using Attack Graphs," in Military Communications Conference, Washington, D.C., 2006, pp. 1-10.

[30] S. Nanda and N. Deo, "A Highly Scalable Model for Network Attack Identification and Path Prediction," in SoutheastCon, Richmond, 2007, pp. 663-668.

 ISBN: 978-1-61208-438-1

Reliability-Aware Design Specification for Allowing Reuse-Based Reliability Level Increment

Work in progress

Patricia López
Tekniker
Eibar, Spain
e-mail:patricia.lopez@tekniker.es

Leire Etxeberria, Xabier Elkorobarrutia
Electronics and Computing Department
Mondragon Unibertsitatea, Engineering Faculty
Mondragon, Spain
e-mail:{letxeberria,xelkorobarrutia}@mondragon.edu

Abstract— **The development of safety-critical systems is expensive and reuse can be seen as a way of reducing the development cost of safety-critical systems. In this context, models could be helpful for safety-critical system development and also to facilitate safe reuse. In this paper, an approach for allowing the reuse-based reliability level increment is presented. This approach is based on a holistic reliability-aware design specification which is related to reliability levels using a knowledge base.**

Keywords-Reliability; safety; reuse; model-based .

I. INTRODUCTION

Cyber-Physical Systems (CPS) are embedded ICT systems that are interconnected, interdependent, collaborative, and autonomous. They provide computing and communication, monitoring/control of physical components/processes in various applications including safety critical. Safety is a key aspect of Safety-critical CPSs. A safety-critical CPS is a CPS whose failure or malfunction may result in death or serious injury to people, loss or severe damage to equipment/property or environmental harm.

The cost of developing safety-critical CPSs is much higher than the cost of developing other kind of software. "A commonly accepted rule of thumb is that development of safety-certified software costs roughly 10 times, as much as non-certified software with equivalent functionality" [1]. Moreover, CPSs have usually real-time constraints and this increases the complexity, "the cost of developing safety-critical software is likely to be 20 to 30 times the cost of developing typical management information software" [1].

Evolution of products is also more costly in safety-critical systems as the re-certification may imply very time-consuming re-doing activities such as re-design, re-verification and re-validation.

Reuse can be seen as a way of reducing development (and specially re-development) costs of safety-critical systems. However, reuse is quite challenging in safety-critical domains as safety must be guaranteed.

Safety-critical systems are developed following domain-specific safety standards that rule what kind of techniques must be used depending on the reliability level to be obtained and safety argumentation is made based on a specific context. And reuse implies to change the context or reliability level.

Models could be helpful for safety-critical system development and also for facilitate reuse. Model-Driven Engineering (MDE) refers to the systematic use of models as primary engineering artifacts throughout the engineering lifecycle. The complexity of system engineering is increasing and model-driven engineering helps to deal with this increasing complexity. For the development of safety-critical systems, MDE could be used for different purposes [2][3]:

- MDE-based development of safety-critical systems: MDE used during the development process of systems for development, verification and validation purposes.
- MDE-based safety certification: MDE for managing safety evidences, MDE for supporting the verification of compliance to safety standards, etc.

This paper presents a model based approach for supporting the reuse of safety critical systems with a special focus on facilitating the increment of reliability level when a product is reused.

Section II presents the state of the art in the area, section III presents the Model-based Approach for Reuse-based Reliability level Increment, section IV addresses the case study that has been used and to finish the conclusions and future work section.

II. STATE OF THE ART

A. Reuse in safety-critical systems

Reuse in safety-critical systems is a research topic that has received quite attention lately. European projects, such as Safety Certification of Software-Intensive Systems with Reusable Components (SafeCer) or Open Platform for EvolutioNary Certification Of Safety-critical Systems (Opencoss) have been focused on reusability of safety critical systems.

There are different reuse scenarios in safety: Intra-standards when reuse is done in the same domain and to meet the same standard or inter-standard or cross domain

 ISBN: 978-1-61208-438-1

when a component or system is reused in another domain and must meet another standard.

In the intra-standard scenario, the reason of reusing could be also different: evolutionary scenario when a system or component changes and we need to assure that is safe, a new product with slightly different requirements, a family of products, when the standard evolve (new version of the standard), when we want to increment the reliability level, etc.

Different kinds of artifacts could be reused as well: requirements [4], components [5], system, safety argumentation, safety case [6][7][29], hazard analysis [8][9]… Depending on what is reused, the phase of the life cycle where is reused is also different; mainly two broad phases could be distinguished: Reuse during construction of the system according to the safety requirements or Reuse during accreditation and certification of the system: providing evidence.

B. Reliability levels

"Traditionally, certification standards have been process-oriented, i.e., where a hazard analysis is performed to identify the severity and risks associated in functional failure for determining a Safety Level, which in turn is used to choose and customize the process applied" [10]. This safety level specifies a target level of risk reduction.

These safety or reliability levels are different depending on the domain-specific standard. IEC 61508 standard, who is intended to be a basic functional safety standard applicable to all kinds of industry, defines the Safety integrity levels (SIL). There are four discrete integrity levels associated with SIL with SIL 4 the most dependable and SIL 1 the least. The SIL can be assigned to any safety relevant function or system or sub-system or component. The

The SIL allocation is made taking into account the rate of dangerous failures and tolerable hazard rate of the function, system, sub-system or component. In the standard each SIL level is associated to a set of measures to be implemented into the design during the design process.

The standards derived from IEC 61508 such as the standards for industrial processes (IEC 61511), or railway industry (EN 50126/EN 50128 /EN 50129) also use SIL.

Other standards specified other levels. In the automotive domain (ISO 26262), the Automotive Safety Integrity Level (ASIL) is used, a risk classification scheme that is an adaptation of the SIL for the automotive industry. The ASIL is established by performing a risk analysis of a potential hazard by looking at the Severity, Exposure and Controllability of the vehicle operating scenario. The safety goal for that hazard in turn carries the ASIL requirements. There are four ASILs identified by the standard: ASIL A, ASIL B, ASIL C, ASIL D. ASIL D dictates the highest integrity requirements on the product and ASIL A the lowest.

For airborne systems (the DO-178C and DO-254 standards) Design Assurance Levels (DAL) are proposed. The DAL is determined from the safety assessment process and hazard analysis by examining the effects of a failure condition in the system. There are five levels of compliance, A through E, which depend on the effect a failure will have on the operation of the aircraft. Level A is the most stringent, defined as "catastrophic" (e.g., loss of the aircraft), while a failure of Level E will not affect the safety of the aircraft.

The different kind of levels could be compared as they have some similarities, but they have also differences; there is not a one-to-one mapping.

Apart from standards, at OPENCOSS project they have developed the concept of Assured reliability and Resilience Level (ARRL) of components [11]. It is an approach that is not applied at system level but at component level, which helps to compose safe systems from components. It is based on the Quality of Service of a component, which is a more generic criterion that takes the trustworthiness as perceived by users better into account. This concept complements the Safety Integrity Level concept.

C. Reliability or Certification-aware design specification

As stated in [12] "*Unfortunately, little work has been done to date on accommodating the additional demands that certification imposes on how the design of systems should be expressed. Our experience indicates that certification is often (incorrectly) viewed as an after-the-fact activity. This can give rise to various problems during certification, because a large fraction of the safety evidence necessary for certification has to be gathered during the design phase and embodied in the design specification. Failing to make the design "certification-aware" will inevitably lead to major omissions and effectively make the design "unauditable" for certification purposes.*"

In [12], they propose a methodology and guidelines for modeling Software-Hardware Interfaces using SysML (Block Definition Diagrams, Internal Block Diagrams, Activity Diagrams and Requirement Diagrams). The goal is to describe the design and establish the traceability (link requirements and design).

Although [12] introduced the concept of "certification-aware design specification" and proposed a methodology, not all the aspects needed to get a reliability-aware design specification are covered. To the best of our knowledge, there is not a holistic approach for specifying a reliability-aware specification.

This design specification should be Product-aware and also Process-aware. Product-aware specification should provide aspects, such as requirements-design traceability, test case-requirements traceability, the applied fault tolerance techniques reflected in the design, failure modes linked to design elements, properties and contracts (formal methods) linked to design elements and requirements…

The process-aware specification should include information about the safety standards that have been applied, the reliability level, the used techniques in the phases of the life cycle and the link to the results of the applied techniques (some aspects specified in the product-aware part, testing results, results of formal proofs…).

There are approaches that cover part of the needs of a holistic reliability-aware design specification:

For requirement analysis and modeling requirement traceability [12][13][14][15][16], etc. For adding formal

properties or contracts to the specification: [17][18]. There are a lot of approaches for relating safety analysis concepts and design specification or transforming the design in safety analysis concepts: [19][20][21][22], etc. For specifying fault tolerance techniques, safety patterns could be used [23] presents an approach for representing Safety Patterns in a design. Regarding Process aware specification, [24] presents a domain model of IEC 61508 concepts: Domain model for SIL activities, Domain model for certification, Domain model for communication, etc. And [25][26] present a conceptual model of evidences for safety cases.

III. MODEL-BASED APPROACH FOR REUSE-BASED RELIABILITY LEVEL INCREMENT

Our approach is based on the following hypothesis:

- To provide a "reliability-aware" design specification helps to reason about the reliability level of a system or component. This can facilitate certification process, the reuse of components and reliability increasing process.
- It is possible to define reliability levels of components and systems and relate this reliability levels to techniques applied during design. Therefore, it is possible to define a decision system that helps to decide which techniques to apply to increase reliability.

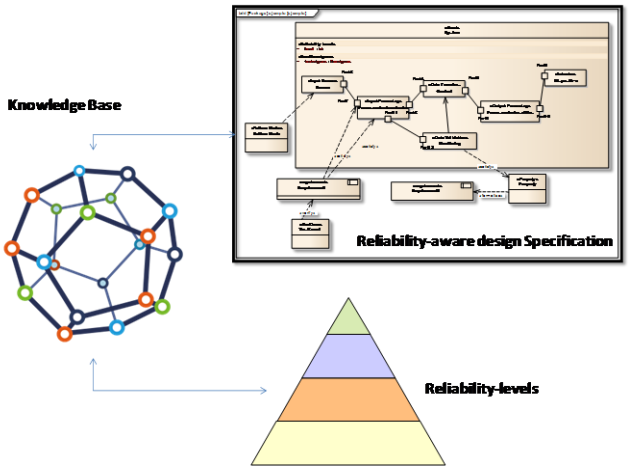


Figure 1. Architecture of the approach

The approach proposes to use a reliability-aware design specification in combination with a reliability-level classification and a knowledge base that relates the levels and the techniques applied and modeled in the specification (see Figure 1).

The main goal of the approach is to facilitate the increment of the reliability level of a system. In industry often it is required to develop a new system with same functionalities as a previous one but with a higher reliability level. The approach will help to reuse the design, verification, validation and certification artifacts of the existing system to a point avoiding expensive re-design and re-certification activities from scratch.

A. *Reliability-aware design specification*

The proposed specification is based on SysML and existing approaches has been reviewed, selected and combined to support a holistic reliability-aware view. System Modeling Language (SysML) is a graphical modeling language for System Engineering. It can be considered as an extension of UML2 for systems. It supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities. SysML is a Critical Enabler for Model Driven System Engineering. SysML could be considered the de-facto standard for systems engineering [27]. Moreover, SysML is rapidly becoming the notation of choice for developing safety-critical systems [13].

The specification has two differentiated parts: the Product-aware specification and the Process-aware specification.

For the Product-aware specification, the following aspects are modeled:

Structural modeling is done using Block Definition Diagrams (bdd) and Internal block diagrams (ibd) of SysML. SysML employs the concept of blocks to specify hierarchies and interconnection within a system design. A BDD describes the system hierarchy and system/component classifications; it lets you describe relationships between blocks, such as composition, association, and specialization. Whereas the IBD describes the internal structure of a system in terms of its parts, ports, and connectors. Interfaces are described using the Port concept of ibds.

For *requirements*, the SysML Requirements diagram is used. Requirements diagram is an extension of the class diagram that allows the modelling of detailed system requirements. It represents the system requirements and their relationships. *Traceability* links are gathered in the diagram: among requirements, among requirements and test cases, among requirements and design and among requirements and other model elements (use cases…). *Test cases* are modeled as special blocks with <<Test Case>> stereotypes to allow the traceability to requirements and design blocks.

Formal properties proven using formal methods are also modeled using an adaptation of the proposal of [17]. Properties are traced to design elements and requirement blocks.

Fault tolerance techniques such as monitors or replication are modeled using safety patterns [23].

And design elements are trace to *failure modes.*

The Process-aware specification includes:

- The *reliability level* assigned to the component or system
- The *standard* applied
- The *list of techniques* applied in each phase
- And *links to the product-aware part* and *results* (testing results, results of formal proofs…).

 ISBN: 978-1-61208-438-1

Meta-data in the Sysml model is used for specifying process-aware information for example using attributes with stereotypes in a block (see figure 2).

B. *Reliability levels*

Reliability levels will be defined for components and systems. This will be done based on ARRL [11] as it provides the reliability level at component level and SIL levels.

C. *Decision system for increasing reliability*

A decision system is being developed that will support reuse, especially increasing reliability level of a component/system.

Based on the reliabilility-aware design specification is possible to know the applied techniques and results and assign a reliability level.

A knowledge base will be developed for being able to relate reliability levels and techniques and guide the increment of reliability. This base will help to answer the following questions:

- Which techniques should be applied to increase reliability?
- Which is the current level of reliability of a design?
- …

IV. CASE STUDY

The approach is being applied to a case study. As first case study, an educational use case has been selected [28]. This educational demonstrator has been previously used in lectures related to safety, real-time, software engineering and embedded system development. It is based on an elevator system control. The elevator system is composed of 2 or more elevators and they lift or bring down a load in a coordinated way. Each elevator has attached a motor, up and down sensor and shaft rotation sensor that is used to infer position and speed.

Each elevator is controlled by an ElevatorCtrl software component. It reads from its sensor, actuates on its motor and announces its state to the main controller. All elevator coordination is in charge of ElevatorSystemCtrl. The one that commands all the elevators on response to an operator. The operator has an interface for commanding the system.

The system is assigned next safety requirements:

- If one crane/elevator stops, the others must stop within 50 millisecond.
- The difference of position between two elevators can't be greater than 10 mm.

Depending on the context where this system will be used, the required reliability level could vary. A first version of the design has been specified using the reliability-aware design specification.

This specification gathers the design of the system (components, interfaces, ports, etc.) using SysML. The traceability information has been also captured: requirements traced to other requirements (some requirements are derived from the "If one elevator stops, the others must stop within 50 milliseconds" requirements), requirements traced to the test cases that verify the requirement and requirements trace to the design elements (components, ports…) that satisfy the requirement. Formal properties such as safety contracts that have been used for verification of timing have been also specified.

Safety patterns applied to the design are showed explicitly such as the Monitor pattern of the Communication Supervisor used in the system.

Finally, metadata is used to add information about the process such as the required reliability level and the used techniques.

The figure 2 is an excerpt of the reliability-aware design specification (concepts that were captured in different diagrams have been mixed for presentation purposes).

One of the benefits of having a reliability-aware design specification is that it facilitates the reuse of the system's design, the reuse of verification & validation artifacts and also the reuse of certification artifacts.

As next step, a scenario of reusing with an increment of reliability level is foreseen.

V. CONCLUSIONS AND FUTURE WORK

First results, especially of the reliability-aware design specification show interesting findings. The approach could be useful for reusing the design with different purposes not only for incrementing reliability. Moreover, the approach is also useful for novel safety engineers or companies that start developing safety-critical systems but they have not so much experience with standards.

However, we have only preliminary results with an educational case study. Further work is needed to see the applicability of the approach in industry.

 ISBN: 978-1-61208-438-1

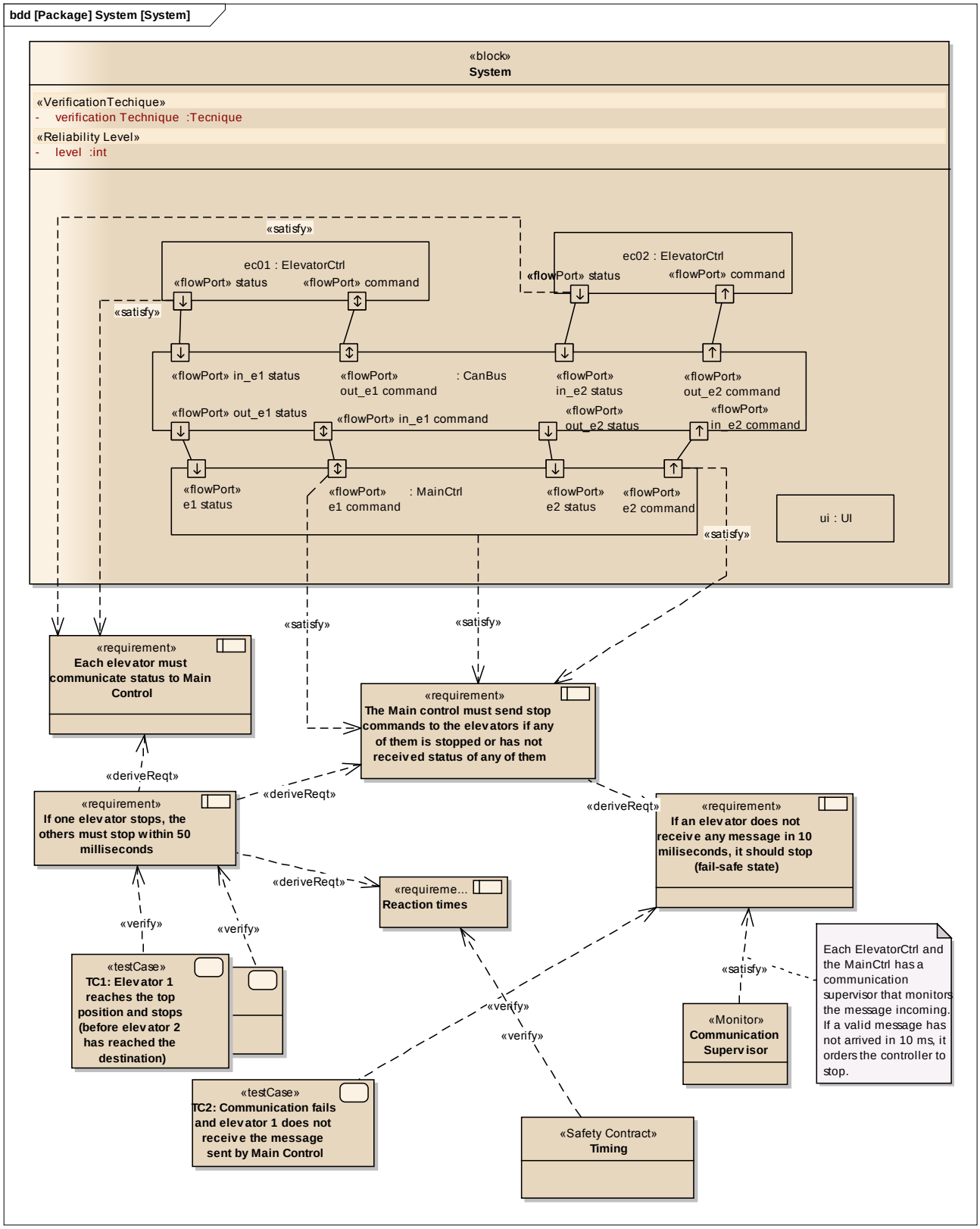


Figure 2. Excerpt of the reliability-aware design specification of the case study

 ISBN: 978-1-61208-438-1

REFERENCES

[1] K. Nilsen, Certification Requirements for Safety-Critical Software, RTC magazine, 2004, http://www.rtcmagazine.com/articles/view/100010, retrieved: October, 2015.

[2] J. L. de la Vara et al, Towards a model-based evolutionary chain of evidence for compliance with safety standards. In "Proceedings of the 2012 international conference on Computer Safety, Reliability, and Security (SAFECOMP'12)", F. Ortmeier and P. Daniel (Eds.). Springer-Verlag, Berlin, Heidelberg, 2012, pp.64-78.

[3] R. K. Panesar-Walawege, Using model-driven engineering to support the certification of safety-critical systems, Doctoral thesis, University of Oslo, 2012

[4] J. Dehlinger and R.R. Lutz. 2005. A product-line requirements approach to safe reuse in multi-agent systems. "SIGSOFT Softw. Eng. Notes" 30, 4, 2005,pp. 1-7.

[5] R. Land, M. Åkerholm, and J. Carlson. 2012. Efficient software component reuse in safety-critical systems - an empirical study. In "Proceedings of the 31st international conference on Computer Safety, Reliability, and Security (SAFECOMP'12)", F. Ortmeier and P. Daniel (Eds.). Springer-Verlag, Berlin, Heidelberg, 2012, pp. 388-399.

[6] P. Fenelon, T. P. Kelly, and J. A. McDermid, Safety Cases for Software Application Reuse. In the "proceedings of SAFECOMP '95", 1995, pp. 419-436

[7] D. Bush, Towards Formalising Reuse in Safety Cases, "Proceedings of the INCOSE UK Spring Symposium", Tolleshunt Knights, Essex, 2002.

[8] S. Baumgart, Investigations on hazard analysis techniques for safety critical product lines, "IDT Workshop on Interesting Results in Computer Science and Engineering (IRCSE)", 2012.

[9] L. Grunske, B. Kaiser, and R. H. Reussner, Specification and evaluation of safety properties in a component-based software engineering process. In "Component-Based Software Development for Embedded Systems", C. Atkinson, C. Bunse, H. G. Gross, and C. Peper (Eds.). Springer-Verlag, Berlin, Heidelberg, 2005, pp. 249-274.

[10] SafeCer project (Safety Certification of Software-Intensive Systems with Reusable Components), http://safecer.eu/, retrieved: October, 2015.

[11] E. Verhulst and B. H. C. Sputh, , ARRL: A criterion for compositional safety and systems engineering: A normative approach to specifying components, "Software Reliability Engineering Workshops (ISSREW)", 2013 IEEE International Symposium on , vol., no., 4-7 Nov. 2013, pp.37-44

[12] M. Sabetzadeh, S. Nejati, L. Briand, and A. H. Evensen Mills, Using SysML for Modeling of Safety-Critical Software-Hardware Interfaces: Guidelines and Industry Experience. In Proceedings of the 2011 IEEE 13th International Symposium on High-Assurance Systems Engineering (HASE '11). IEEE Computer Society, Washington, DC, USA, 2011, pp.193-201.

[13] S. Nejati, M. Sabetzadeh, D. Falessi, L. Briand, and T. Coq. 2012. A SysML-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies." Inf. Softw. Technol." 54, 6, 2012, pp. 569-590.

[14] D. Falessi, S. Nejati, M. Sabetzadeh, L. Briand, and A. Messina, SafeSlice: a model slicing and design safety inspection tool for SysML. In "Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE '11)". ACM, New York, NY, USA, 2011, pp. 460-463.

[15] A. Albinet, J.-L. Boulanger, H. Dubois, M.-A. Peraldi-Frati, Y. Sorel, and Q.-D. Van, Model-based methodology for requirements traceability in embedded systems, in "Proceedings of 3rd European Conference on Model Driven Architecture Foundations and Applications, ECMDA'07", Haifa, Israel, 2007.

[16] P. Colombo, F. Khendek, and L. Lavazza, Requirements analysis and modeling with problem frames and SysML: a case study. In "Proceedings of the 6th European conference on Modelling Foundations and Applications (ECMFA'10)", T. Kühne, B. Selic, M.-P. Gervais, and F. Terrier (Eds.). Springer-Verlag, Berlin, Heidelberg, 2010, pp.74-89.

[17] J.-F. Pétin, D. Evrot, G. Morel, and P. Lamy, Combining SysML and formal models for safety requirements verification, "ICSSEA 2010", 2010.

[18] S. Tonetta, Contract-based design of safety-critical software components, "International Workshop on Critical Software Component Reusability and Certification across Domains (CSC 2013)", ICSR13 workshop, June 18 2013

[19] K. Thramboulidis and S. Scholz, Integrating the 3+1 SysML view model with safety engineering, "Emerging Technologies and Factory Automation (ETFA)", 2010 IEEE Conference on , vol., no., 1,8, 2010, pp.13-16

[20] G. Li and B. Wang, SysML aided safety analysis for safety-critical systems. In "Proceedings of the Third international conference on Artificial intelligence and computational intelligence - Volume Part I (AICI'11)", H. Deng, D. Miao, J. Lei, and F. L. Wang (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 2011, pp. 270-275.

[21] F. Mhenni, N. Nguyen, H. Kadima, and J. Choley, Safety analysis integration in a SysML-based complex system design process, "Systems Conference (SysCon)", 2013 IEEE International, vol., no., 2013, pp.70-75

[22] J. Xiang, K. Yanoo, Y. Maeno, and K. Tadano, Automatic Synthesis of Static Fault Trees from System Models, "Secure Software Integration and Reliability Improvement (SSIRI)", 2011 Fifth International Conference on, 2011, pp.127-136

[23] P. Antonino, T. Keuler, E.Y. Nakagawa, , Towards an approach to represent safety patterns, "The Seventh International Conference on Software Engineering Advances, ICSEA", 2012.

[24] D. Kuschnerus, F. Bruns, T. Musch, A UML Profile for the Development of IEC 61508 Compliant Embedded Software, "Embedded Real Time Software and Systems - ERTS² ", 2012.

[25] R. K. Panesar-Walawege, M. Sabetzadeh, and L. Briand, Using UML Profiles for Sector-Specific Tailoring of Safety Evidence Information, "30th International Conference, ER 2011", Brussels, Belgium, October 31 - November 3, 2011, pp.362-378

[26] R. K. Panesar-Walawege, M. Sabetzadeh,L. Briand, T. Coq, Characterizing the Chain of Evidence for Software Safety Cases: A Conceptual Model Based on the IEC 61508 Standard, "Software Testing, Verification and Validation (ICST)", 2010 Third International Conference on , vol., no., 6-10 April 2010,pp.335-344

[27] W. Schafer, and H. Wehrheim, The Challenges of Building Advanced Mechatronic Systems, "Future of Software Engineering, FOSE '07", 23-25 May 2007, pp.72-84

[28] M. Illarramendi, L. Etxeberria, and X. Elkorobarrutia, Reuse in Safety Critical Systems: Educational Use Case First Experiences. In "Proceedings of the 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA '14)". IEEE Computer Society, Washington, DC, USA, 2014, pp. 417-422.

[29] I. Sljivo, Facilitating Reuse of Safety Case Artefacts Using Safety Contracts, Doctoral thesis, Mälardalen University, 2015

 ISBN: 978-1-61208-438-1

Best Practices for the Design of RESTful Web Services

Pascal Giessler
and Michael Gebhart

iteratec GmbH
Stuttgart, Germany
Email: pascal.giessler@iteratec.de,
Email: michael.gebhart@iteratec.de

Dmitrij Sarancin, Roland Steinegger,
and Sebastian Abeck

Cooperation & Management
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Email: dmitrij.sarancin@student.kit.edu,
Email: roland.steinegger@kit.edu,
Email: sebastian.abeck@kit.edu

***Abstract*—The trend towards creating web services based on the REpresentational State Transfer (REST) is unbroken. Because of this, several best practices for designing RESTful web services have been created in research and practice to ensure a certain level of quality. But, these best practices are often described differently with the same meaning due to the nature of natural language. In addition, they are not collected and presented in a central place but rather distributed across several pages in the World Wide Web, which impedes their application even further. In this article, we identify, collect, and categorize several best practices for designing RESTful web services and illustrate their application on a real system to show their application. For illustration purpose, we apply the best practices on the CompetenceService, an assistance service of the SmartCampus system developed at the Karlsruhe Institute of Technology (KIT).**

Keywords–REST; RESTful; best practices; collection; catalog; design; quality; research and practice

I. INTRODUCTION

Over the years, more and more web services based on the architectural style REST were developed, which uses existing functionality from the application layer protocol Hypertext Transfer Protocol (HTTP) [1] [2]. This results in an increasing interest compared to traditional web services with Simple Object Access Protocol (SOAP), which can be shown in a Google Trend Analysis with the keywords *REST* and *SOAP* or in the increasing usage of REST- instead of SOAP-based web services [1]. Also big companies, such as Twitter or Amazon, are using REST-like interfaces for their services, which are shown in their Application Programming Interface (API) documentations.

Despite this trend, there are still no standards or guidelines about how to develop a RESTful web service. Instead of this, several best practices in research and practice have been developed and were published in a range of articles, magazines and pages in the World Wide Web (WWW). But, these best practices were often described differently with the same semantics due to the nature of natural language [3]. This results in several obscurities and misconceptions by applying these best practices.

To overcome these issues, we have collected, categorized and formalized several best practices in a way that they can be easily applied during the development of RESTful web services , as well as for analyzing existing RESTful web services. More precisely, we have defined eight different categories and found an amount of 23 best practices that will be described in this paper. These best practices provide guidelines for the design of RESTful web services to support certain quality goals such as the usability of the Web API. Furthermore, their usage also results in an increasing consistency of web services.

For illustration, we have used this set of best practices for the development of the CompetenceService as part of the SmartCampus system at the KIT. The SmartCampus is a system which provides functionality for students, guests and members of an university to support their daily life. Today, the SmartCampus already offers some services, such as the ParticipationService to support the decision-making process between students, professors and members of the KIT with a new approach called system-consenting [4]. The developed services at the SmartCampus are based on REST, so that they can be used for several different devices as a lightweight alternative to SOAP.

The current paper is structured as follows: In Section II, the architectural style REST will be described in detail to lay the foundation for this paper. Afterwards, existing papers and articles will be discussed in Section III to show the necessity of identification, collection, and categorization of existing best practices for RESTful web services. The CompetenceService is used to illustrate the best practices will be presented in Section IV. In Section V, the best practices for RESTful web services will be presented in detail so that they can be easily applied during the design phase of such web services. Finally, a summary of this paper and an outlook on further work will be given in Section VI.

II. FOUNDATION

REST is an architectural style, which was developed and first introduced by Fielding [5] in his dissertation. According to Garlan and Shaw [6], an architectural style can be described as follows: “an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined.” [6, p. 6].

 ISBN: 978-1-61208-438-1

For the design of REST, Fielding [5] has identified four key characteristics, which were important for the success of the current WWW [7]. To ensure these characteristics, the following constraints were derived from existing network architectural styles together with another constraint for the uniform interface [5]: 1) Client and Server, 2) Statelessness, 3) Layered Architecture, 4) Caching, 5) Code on Demand and 6) Uniform Interface. The latter one represents the Web API of RESTful web services and can be seen as an umbrella term, since it can be decomposed into four sub-constraints [7]: 6.1) Identification of resources, 6.2) Manipulation of resources through representations, 6.3) Self-descriptive messages and 6.4) Hypermedia.

If all of these constraints are fulfilled by a web service, it can be called RESTful. The only exception is "Code on Demand", since it is an optional constraint and has not to be implemented by a web service.

III. State of the Art

This section discusses different articles, magazines and approaches in the context of RESTful best practices, which respect the architectural style REST and its underlying concepts.

In Fielding [5], Fielding presents the structured approach for designing the architectural style REST, while it remains unclear how a REST-based web service can be developed in a systematic and comprehensible manner. Furthermore, there is also a lack of concrete examples of how hypermedia can be used as the engine of the application state, which can be one reason why REST is understood and implemented differently.

Mulloy [8] presents different design principles and best practices for Web APIs, while he puts the focus on "pragmatic REST". By "pragmatic REST" the author means that the usability of the resulting Web API is more important than any design principle or guideline. But, this decision can lead to neglecting the basic concepts behind REST such as hypermedia.

Jauker [9] summarizes ten best practices for a RESTful API, which represent, in essence, a subset of the described best practices by Mulloy [8] and a complement of new best practices. Comparable with [8], the main emphasis is placed on the usability of the web interface and not so much on the architectural style REST, which can lead to the previously mentioned issue.

Papapetrou [10] classifies best practices for RESTful APIs in three different categories. However, there is a lack of concrete examples of how to apply these best practices on a real system compared to the two previous articles.

In Vinoski [11], a checklist of best practices for developing RESTful web services is presented, while the author explicitly clarifies that REST is not the only answer in the area of distributed computing. He structures the best practices in four sections, which addressing different areas of a RESTful web service such as the representation of resources. Despite all of his explanations, the article lacks in concrete examples to reduce the ambiguousness.

Richardson et. al [7] cover in their book as a successor of [12], among other topics, the concepts behind REST and a procedure to develop a RESTful web service. Furthermore, they place a great value on hypermedia , as well as Hypermedia As The Engine Of Application State (HATEOAS), which is not taken into account by all of the prior articles. But, the focus of this work is the comprehensive understanding of REST rather than providing best practices for a concrete implementation to reduce the complexity of development decisions.

In Burke [13], Burke presents a technical guide of how to develop web services based on the Java API for RESTful Web Services (JAX-RS) specification. But, this work focuses on the implementation phase rather than the design phase of a web service, where the necessary development decisions have to be made.

IV. Scenario

The SmartCampus is a modern web application, which simplifies the daily life of students, guests, and members at the university. Today, it offers several services, such as the ParticipationService for decision-making [4], the SmartMeetings for discussions or the CampusGuide for navigation and orientation on the campus. By using non-client specific technologies, the services can be offered to a wide range of different client platforms, such as Android or iOS.

The CompetenceService is a new service as part of the SmartCampus to capture and semantically search competences in the area of information technology. For easier acquisition of knowledge information, the CompetenceService offers the import of competence and profile information from various social networks such as LinkedIn or Facebook. The resulting knowledge will be represented by an ontology, while the profile information will be saved in a relational database. SPARQL Protocol And RDF Query Language (SPARQL) is used as the query language for capturing and searching knowledge information in the ontology.

In Figure 1, the previously described CompetenceService is illustrated in the form of a component diagram. For implementation of the CompetenceService, the Java framework Spring was used.

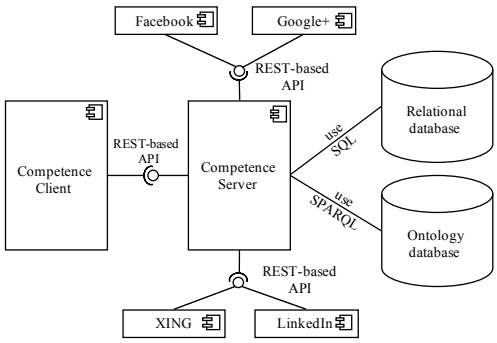


Figure 1. Component model of the CompetenceService.

To demonstrate the benefits of this service, a simple use case will be described in the following. A young startup company is looking for a new employee, who has competences in "AngularJS" and "Bootstrap". For that purpose, the startup company uses the semantics search engine of the CompetenceService to search for people with the desired skills. The resulting list of people will be ordered by relevance so that the startup company can easily contact the best match.

 ISBN: 978-1-61208-438-1

V. BEST PRACTICES FOR RESTFUL WEB SERVICES

This section presents eight different categories of best practices for designing RESTful web services, whereby each one is represented by a subsection. The contained best practices have to be seen as recommendations to design and improve such services rather than as strict guidelines. So, it is fine, if not all of the given best practices are fulfilled by a RESTful web service so long as an understandable reason for not considering one can be given. Furthermore, it is important to point out here that the fulfillment of the following best practices does not guarantee the compliance of the mentioned constraints in Section II. For this, the Richardson Maturity Modell (RMM) can be used to analyze the preconditions of a RESTful web service [14].

A. No Versioning

Versioning of a Web API is one of the most important considerations during the design of web services since the API represents the central access point of a web service and hides the service implementation. This is why a web interface should never be deployed without any versioning identifier according to Mulloy [8]. For versioning, many different approaches exist such as embedding it into the base Uniform Resource Identifier (URI) of the web service or using the HTTP-Header for selecting the appropriate version [8]. But, web services based on REST do not need to be versioned due to hypermedia.

That is why, RESTful web services can be compared with traditional websites that are still accessible on all web browsers when modifying the content of the websites. So, no additional adjustment is necessary on the client side. Furthermore, versioning also has a negative impact on deployed web services, which Fielding states as follows: "Versioning an interface is just a polite way to kill deployed applications" [15] since it increases the effort for maintaining the web service.

B. Description of resources

The description of resources correlates with the usability of the web service since the resources represent or abstract the underlying domain model. For this category, five best practices could be identified:

1) According to Vinoski [11], Papapetrou [10] and Mulloy [8], nouns should be used for resource names.
2) The name of a resource should be concrete and domain specific, so that the semantics can be inferred by a user without any additional knowledge [8] [10].
3) The amount of resources should be bounded to limit the complexity of the system, whereby this recommendation depends on the degree of abstraction of the underlying domain model [8].
4) The mixture of plural and singular by naming resources should be prevented to ensure consistency [8] [9].
5) The naming convention of JavaScript should be considered since the media type JavaScript Object Notation (JSON) is the most used data format for the client and server communication by this time [2] [8] [16].

Figure 2 illustrates the first, second and third best practice of this category.

```
1  /* ProfileController */
2  @RestController
3  @RequestMapping(value = "/profiles")
4  public class ProfilesController {
5    ...
6    @RequestMapping(method = RequestMethod.GET)
7    public List<Profile> getProfiles() {...}
8    ...
9  }
10
11 /* CompetenceController */
12 @RestController
13 @RequestMapping(value = "/competences")
14 public class CompetenceController {
15   ...
16   @RequestMapping(method = RequestMethod.GET)
17   public List<Competence> getCompetences() {...}
18   ...
19 }
```

Figure 2. Example for description of resources.

C. Identification of Resources

According to Fielding [5], URIs should be used for unique identification of resources. For this constraint, we have found four best practices:

1) An URI should be self-explanatory according to the affordance [8]. The term affordance refers to a design characteristic by which an object can be used without any guidance.
2) A resource should only be addressed by two URIs. The first URI address represents a set of states of the specific resource and the other one a specific state of the previously mentioned set of states [8].
3) The identifier of a specific state should be difficult to predict [10] and not references objects directly according to the Open Web Application Security Project (OWASP) [17], if there is no security layer available.
4) There should be no verbs within the URI since this implies a method-oriented approach such as SOAP [8] [9].

Figure 3 illustrates the second best practice of this category. Note that there are no verbs within the URIs, hence the fourth best practice is also fulfilled.

```
1 /* Set of profiles */
2 competence-service/profiles
3
4 /* Specific profile with identifier {id} */
5 competence-service/profiles/{id}
```

Figure 3. Example for identification of resources.

D. Error Handling

As already mentioned, the Web API represents the central access point of a RESTful web service, which is comparable with a provided interface of a software component [18]. Each information about the implementation of the service is hidden by the interface. Therefore, only the outer behavior can be

 ISBN: 978-1-61208-438-1

observed through responses by the web service, which is why well-known software debugging techniques such as setting exception breakpoints can not be applied.

For this reason, the corresponding error message has to be clear and understandable so that the cause of the error can be easily identified. With this in mind, we could identify three best practices:

1) The amount of used HTTP status codes should be limited to reduce the feasible effort for looking up in the specification [8] [9].
2) Specific HTTP status codes should be used according to the official HTTP specification [19] and the extension [20] [9] [11] [10].
3) A detailed error message should be given as a hint for the error cause on client side [8] [9]. That is why, an error message should consist of four ingredients: 3.1) a message for developers, which describes the cause of the error and possibly some hints how to solve the problem, 3.2) a message that can be shown to the user, 3.3) an application specific error code and 3.4) a hyperlink for further information about the problem.

Figure 4 illustrates the mentioned ingredients of an error message according to the third best practice of error handling.

```
1  HTTP/1.1 404 NOT FOUND
2  /* More header information */
3  {
4    "error" : {
5      "responseCode" : 404,
6          "errorCode" : 107,
7          "messages" : {
8            "developer" : "The resource 'profile'
                 could not be found.",
9            "user" : "An error occurred while
                 requesting the information. Please
                  contact our technical support."
10         },
11         "additionalInfo": ".../competence-
               service/errors/107"}
12 }
```

Figure 4. Example for detailed error message.

E. Documentation of the Web API

A documentation for Web APIs is a debatable topic in the context of RESTful web services since it represents an out-of-band information, which should be prevented according to Fielding: "Any effort spent describing what method to use on what URIs of interest should be entirely defined within the scope of the processing rules for a media type" [21]. This statement can be explained with the fact that documentation is often used as a reference book in traditional development scenarios. As a result of this, it can lead to hardcoded hyperlinks in the source code instead of interpreting hyperlinks of the current representation following the HATEOAS principle. Also business workflows will be often implemented according to the documentation. In this case, we call it Documentation As The Engine Of Application State (DATEOAS). As a result of this, we have developed a new kind of documentation in consideration of HATEOAS to give developers a guidance for developing a client component.

The new documentation consists of three ingredients: 1) Some examples which show how to interact with different systems according to the principle of HATEOAS due to the fact that some developers are not familiar with this concept [21], 2) an abstract resource model in form of a state diagram, which defines the relationship and the state transitions between resources. Also a semantics description of the resource and its attributes should be given in form of a profile such as Application-Level Profile Semantics (ALPS) [22], which can be interpreted by machines and humans and 3) a reference book of all error codes should be provided so that developers can get more information about an error that has occurred.

Figure 5 illustrates an abstract resource model of the CompetenceService. Based on this model, it can be derived which request must be executed to get the desired information. For example to get all competences of a specific profile, we have to first request the resource *profiles*. This results in a set of available profiles, whereby each profile contains one hyperlink for further information. After following the hyperlink by selecting the desired profile, the whole information about the profile will be provided , as well as further hyperlinks to related resources such as *competences*.

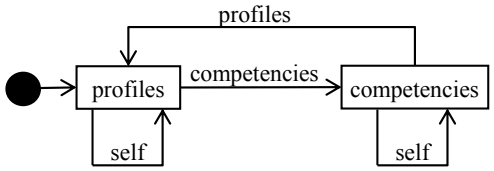


Figure 5. Example for documentation of the Web API.

F. Usage of Parameters

Each URI of a resource can be extended with parameters to forward optional information to the service. In the following, we are focusing on four different use cases since they will be supported by several web services offered by Facebook or Twitter.

1) Filtering: For information filtering of a resource either its attributes or a special query language can be used. The election for one of these two variants depends on the necessary expression power of the information filtering. Figure 6 illustrates how a special user group can be fetched by using a query language [9].

```
1 GET /profiles?filter=(competencies=java%20and%20
      certificates=MCSE_Solutions_Expert)
```

Figure 6. Filtering information by a using query language.

2) Sorting: For information sorting, Jauker [9] recommends a comma separated list of attributes with "sort" as the URI parameter followed by a plus sign as a prefix for an ascending order or a minus sign for a descending order. Finally, the order of the attributes represents the sort sequence. Figure 7 illustrates how information can be sorted by using the attributes *education* and *experience*.

 ISBN: 978-1-61208-438-1

```
1 GET /profiles?sort=-education,+experience
```

Figure 7. Sorting a resource by using attributes.

3) Selection: The selection of information in form of attributes reduces the transmission size over the network by responding only with the requested information. For this purpose, Mulloy [8] and Jauker [9] recommend a comma separated list of attributes and the term *fields* as the URI parameter. Figure 8 represents an example how the desired information can be selected before transmitting over the network.

```
1 GET /profiles?fields=id,name,experience
```

Figure 8. A selection of resource information.

4) Pagination: Pagination enables the splitting of information on several virtual pages, while references for the next (*next*) and previous page (*prev*) exist , as well as for the first and last page (*first* and *last*). As URI parameter, *offset* and *limit* were recommended, whereby the first one identifies the virtual page and the last one defines the amount of information on the virtual page [8] [9]. A default value for *offset* and *limit* can not be given since it depends on the information to be transmitted to the client, which Mulloy stated [8] as follows: "If your resources are large, probably want to limit it to fewer than 10; if resources are small, it can make sense to choose a larger limit" [8, p. 12]. Figure 1 illustrates a request using pagination on the resource *profiles*.

```
1 GET /profiles?offset=0&limit=10
```

Listing 1. Requesting 10 profiles by using pagination.

G. Interaction with Resources

By using REST as the underlying architectural style of a system, a client interacts with the representations of a resource instead of using it directly. The interaction between client and server is built on the application layer protocol HTTP, which already provides some functionality for the communication. For the interaction with a resource, we could identify three different best practices:

1) According to Jauker [9] and Mulloy [8], the used HTTP methods should be conform to the method's semantics defined in the official HTTP specification. So, the HTTP-GET method should only be used by idempotent operations without any side effects. For a better overview, Table I sums up the most frequently used HTTP methods and their characteristics. These characteristics can be used to associate the HTTP methods with the correct Create Read Update Delete (CRUD)-operation [11].
2) The support of HTTP-OPTIONS is recommended if a large amount of data has to be transmitted since it allows a client to request the supported methods of the current representation before transmitting information over the shared medium. But, this additional HTTP-OPTIONS request is only necessary, if the supported operations were not written explicitly in the representation.
3) The support of conditional GET should be considered during the development of a service based on HTTP since it prevents the server from transmitting previously sent information. Only if there are modifications of the requested information since the last request, the server responds with the latest representation. For the implementation of conditional GET, there are two different approaches that are already described by Vinoski [11].

TABLE I. CHARACTERISTICS OF THE MOST USED HTTP METHODS.

HTTP method	safe	idempotent
POST	No	No
GET	Yes	Yes
PUT	No	Yes
DELETE	No	Yes

H. Support of MIME Types

Multipurpose Internet Mail Extensions (MIME) types are used for the identification of data formats, which will be registered and published by the Internet Assigned Numbers Authority (IANA). These types can be seen as representation formats of a resource. For this category, we could identify the following four best practices:

1) At least two representation formats should be supported by the web service, such as JSON or Extensible Markup Language (XML) [8].
2) JSON should be the default representation format since its increasing distribution [8].
3) Existing MIME types should be used, which already support hypermedia such as JSON-LD (JSON for Linking Data), Collection+JSON and Siren [11].
4) Content negotiation should be offered by the web service, which allows the client to choose the representation format by using the HTTP header field "ACCEPT" in his request. Furthermore, there is the opportunity to weight the preference of the client with a quality parameter [11].

VI. SUMMARY AND OUTLOOK

In this article, we identified, collected, and categorized best practices for a quality-oriented design of RESTful web services. More precisely, based on existing work 23 best practices could be identified and classified into eight different categories. The intention of this article was not to reinvent the wheel. For this reason, the best practices of this article were reused from existing work. Focus of the work presented in this article was their collection, categorization, and thus unification. We illustrated the best practices by means of the CompetenceService developed at the KIT. By applying the best practices, the CompetenceService could be designed in a quality-oriented manner. Any time during the design or afterwards, the quality of the design could be systematically evaluated. The clear set of best practices enabled to

 ISBN: 978-1-61208-438-1

perform the evaluation by different developers. Furthermore, the repeatability of the analysis and the comparability of the results were guaranteed. As result, design weaknesses of the CompetenceService could be identified and rapidly corrected and the time spent making design decisions could be reduced.

The best practices and their categorization and unification help software architects and developers to design RESTful web services in a quality-oriented manner. As best practices are distributed across several existing work, until now, a systematic analysis of RESTful web services regarding their design quality has been a complex task. In most cases, software architects and developers have a basic understanding about how to create well-designed web services. However, a common understanding about how to evaluate web services is missing. The unification of best practices introduced in this article reduces the necessity to lookup best practices in literature.

In the future, we plan to investigate the impact of such best practices on the development speed. To evaluate the usefulness of the best practices for RESTful web services, we consider setting up two teams of students, Team A and Team B, with the requirement to develop two services as part of the SmartCampus at the KIT of similar complexity. Both teams are expected to have similar experiences in developing software systems and both teams should not have knowledge about the quality-oriented design of RESTful web services. However, Team A will be equipped with our catalog of best practices for RESTful web services. We expect that Team A will spend much less time searching appropriate design rules and design agreements. The best practices will provide Team A with guidelines about how to design the services. Furthermore, the design of the resulting service supports certain quality goals. However, we expect that the more sophisticated design will result in a more complex implementation phase. Figure 9 shows the expected results.

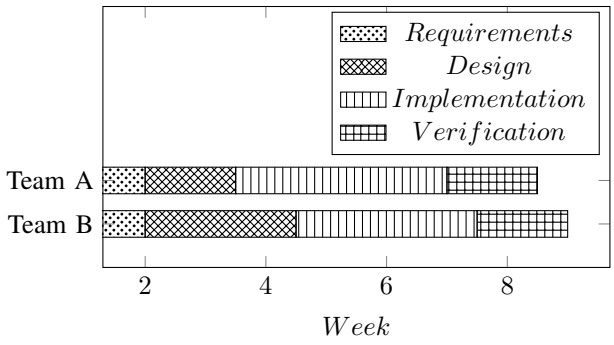


Figure 9. Duration of the development phases in weeks.

In addition, we plan to describe the best practices by means of technology-independent metrics. In a next step, we plan to map these technology-independent metrics onto concrete technologies, such as Java and JAX-RS. This mapping constitutes the basis for an automated application of the metrics on concrete design or implementation artifacts. We are currently working on a software tool, the QA82 Analyzer [23] [24]. This tool enables the automatic evaluation of software artifacts regarding best practices. This tool is available as open source to support the quality-oriented design of RESTful web services in practice, teaching, and research.

REFERENCES

[1] R. Mason, "How rest replaced soap on the web: What it means to you," October 2011, URL: http://www.infoq.com/articles/rest-soap [accessed: 2015-02-20].

[2] A. Newton, "Using json in ietf protocols," the IETF Journal, vol. 8, no. 2, October 2012, pp. 18 – 20.

[3] IEEE, "Std 830-1998: Recommended Practice for Software Requirements Specifications," 1998.

[4] M. Gebhart, P. Giessler, P. Burkhardt, and S. Abeck, "Quality-oriented requirements engineering for agile development of restful participation service," Ninth International Conference on Software Engineering Advances (ICSEA 2014), October 2014, pp. 69 – 74.

[5] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[6] D. Garlan and M. Shaw, "An introduction to software architecture," Pittsburgh, PA, USA, Tech. Rep., 1994.

[7] L. Richardson, M. Amundsen, and S. Ruby, RESTful Web APIs. O'Reilly Media, 2013.

[8] B. Mulloy, "Web API Design - Crafting Interfaces that Developers Love," March 2012, URL: http://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf [accessed: 2015-04-09].

[9] S. Jauker, "10 Best Practices for better RESTful API," Mai 2014, URL: http://blog.mwaysolutions.com/2014/06/05/10-best-practices-for-better-restful-api/ [accessed: 2015-02-19].

[10] P. Papapetrou, "Rest API Best(?) Practices Reloaded," URL: http://java.dzone.com/articles/rest-api-best-practices [accessed: 2015-02-26].

[11] S. Vinoski, "RESTful Web Services Development Checklist," Internet Computing, IEEE, vol. 12, no. 6, 2008, pp. 94–96. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4670126

[12] L. Richardson and S. Ruby, Restful Web Services. O'Reilly Media, 2007.

[13] B. Burke, RESTful Java with JAX-RS 2.0. O'Reilly Media, 2013.

[14] J. Webber, S. Parastatidis, and I. Robinson, REST in Practice: Hypermedia and Systems Architecture. O'Reilly Media, 2010.

[15] R. T. Fielding, "Evolve'13 - The Adobe CQ Community Technical Conference - Scrambled Eggs," 2013, URL: http://de.slideshare.net/royfielding/evolve13-keynote-scrambled-eggs [accessed: 2015-09-23].

[16] A. DuVander, "1 in 5 APIs Say "Bye XML"," 2011, URL: http://www.programmableweb.com/news/1-5-apis-say-bye-xml/2011/05/25 [accessed: 2015-02-20].

[17] OWASP, "Testing for insecure direct object references (otg-authz-004)," 2014, URL: https://www.owasp.org/index.php/Testing_for_Insecure_Direct_Object_References_(OTG-AUTHZ-004) [accessed: 2015-05-12].

[18] J. Bosch, Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach, ser. ACM Press Series. Addison-Wesley, 2000.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Rfc 2616, hypertext transfer protocol – http/1.1," http://tools.ietf.org/html/rfc2616, 1999.

[20] M. Nottingham and R. Fielding, "Rfc 6585, additional http status codes," 2012, URL: http://tools.ietf.org/html/rfc6585 [accessed: 2015-02-18].

[21] R. T. Fielding, "REST APIs must be hypertext-driven," October 2008, URL: http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven [accessed: 2015-02-20].

[22] M. Amundsen, L. Richardson, and M. W. Foster, "Application-Level Profile Semantics (ALPS) ," Tech. Rep., August 2014, URL: http://alps.io/spec/ [accessed: 2015-04-09].

[23] M. Gebhart, "Query-based static analysis of web services in service-oriented architectures," International Journal on Advances in Software, 2014, pp. 136 – 147.

[24] QA82, "QA82 Analyzer," 2015, URL: http://www.qa82.org [accessed: 2015-02-27].

 ISBN: 978-1-61208-438-1

Criteria of Evaluation for Systems Using Sensor as a Service

A discussion about repositories and search engines on S²aaS systems

Anderson Brasiliense de Oliveira Brito
Diretoria de Tecnologia da Informação
Instituto Federal do Amapá, IFAP
Macapá, Brasil
Email: anderson@ifap.edu.br

Felipe Silva Ferraz
Centro de Informática
Universidade Federal de Pernambuco, UFPE
Recife, Brasil
Email: fsf3@cin.ufpe.br

***Abstract*—This document proposes an approach to the criteria settings for evaluation of systems that use sensors as a service through the analysis of sensor repository and search engine data. The sensor paradigm as a service is a branched concept of cloud computing, which is still evolving. Thus, to achieve the end user's expectation for this type of service, it is necessary to define clear parameters for the evaluation of the delivery of it. The proposal presented in this paper provides an analysis based on the type of sensors and how its owner groups them in a multiuser system.**

Keywords- sensor as a service; sensors repository; search engine data.

I. INTRODUCTION

The society has seen an expansion of emerging technologies for the Internet. This phenomenon is a favorable environment for several factors, among which are: the emergence of IPv6 as a protocol with the possibility of enabling many devices connected to a network, the price of sensors, processors and network devices which have their price declined over time, as well as the wireless network for computers, WiFi, which served as input for connecting devices both at home and in offices.

From this scenario, the market expects an environment called "Industrial Internet" will provide 10-15 trillion dollars in the next 20 years. Based on this forecast, it was created an economic value, called "Internet of Everything" until 2020 [1].

The cloud becomes the most suitable environment to support the great mass of devices that will be connected to this global communication network, with a forecast of 50 to 100 billion of connected devices in the aforementioned period [2], and in this scenario, the type node or sensor devices will account for 60% of the total available on the Internet [3].

The delivery of this information is a challenge, considering that there is no standard interface for sensor communication. Another difficulty would be how to manage repository sensors so that it can be qualified and can provide useful data to a system.

The analysis of a sensor repository would enable decision making by the system based on its business model. In an environment of service to the end user where the supply would be the sensor data, the repository qualification will be crucial, because the expectation in the search data held by the user will depend directly of the sensors that the repository provides.

Considering paradigms such as Smart Cities, where the citizen is inserted in this context with the intention to shape innovation and urban development through their participation [4], delivery of these services, which in part can be supplied from sensor data, should take into consideration the repository where such devices are contained. In this sense, mean repository as sensors virtualization that provides data feeds on one or more systems.

This model proved to be very efficient for the market, because companies would not need to invest part of their capital in IT assets, transferring this responsibility and risks to third parties [5].

Some authors [15] propose a modelling for smart cities, in which four layers are defined, namely of 1) sensors and sensor owners, 2) sensor publishers, 3) extended service providers, and 4) sensor date consumers.

The purpose of this article is to provide the analysis of layers 1 and 2. These layers were used as the basis for the delivery service, because they are the basis of data consumption. Without them, the others would not receive data for end users.

Manzoor [16] proposes criteria for quality context QoC, based on the quality information analysis from data obtained from sensors.

Thus, the approach in the definitions of a sensor repository and the data delivery engine will be discussed through evaluation of sensors.

The remainder of this paper is structured as follows. In Section 2, the cloud computing concepts will be addressed. Section 3 will discuss the criteria as well as analyze the results obtained from the queries generated in the sensor repository. Section 4 concludes the paper.

II. CLOUD AND SENSORS

The Cloud Computing began to be broadcast in October 2007, when IBM and Google decided to establish a partnership to create a new model for computing, based on current characteristics of cloud computing with high availability, computational resiliency, resources on demand, from a high quality system [5].

Many authors [5]-[9] group the cloud computing model into three distinct classes, as follows:

 ISBN: 978-1-61208-438-1

- Infrastructure-as-a-Service (IaaS) - class that is characterized with hardware virtualization on the supply side.
- Platform-as-a-Service (PaaS) - the infrastructure is abstracted from a layer between the hardware and applications through an interface feeding. The Azure and Google App platforms are examples of this class.
- Software-as-a-Service (SaaS) - it aims virtualization of local computer applications to the cloud. This class is the highest level of abstraction, getting under the supplier's responsibility to maintain, update and support from both the hardware and the software, leaving the end user only the service consumption.

However, these classes are subject to change or develop their own concepts, because, according to the National Institute of Standards and Technology (NIST) cloud computing is defined as a paradigm in development and thus their definitions, case use, technologies, risks and benefits will be redefined, based on interactions between the public and private sectors [10]. From this view, other terms have been introduced to the academic community. They include the Sensing and Actuation as a Service (SAaaS), Sensor Event as a Service (SEaaS) sensor as a Service (SenaaS), DataBase as a Service (DBaaS) Data as a Service (DaaS), Ethernet as a Service (AAS), Identity and Plicy Management as a Service (IPMaaS) and Video Suveillance as a Service (VSaaS) [11].

Many of these terms have acronyms identical with other terms and that can be confusing. For example, Image as a Service(IaaS) [12] has the same acronym as Infrastructure as a Service and Sensing as a Service (SaaS) [11] has the same acronym as Software as a Service, which is also represented as S²aaS [13]. Thus, a broader class could incorporate all other through the Everything as a Service (XaaS) [14].

III. Definition of Analysis Criteria

From an environment where everything can be offered as a service, cloud enables a favorable site for the sensors expansion through their systems virtualization and subsequent delivery to the end user.

A. Analysis about sensor

To be able to define the repository quality criteria, it will be necessary to assess, first, the characteristics of sensors that compose it. Among the listed characteristics to evaluate a sensor, five were selected and used by [16]:

*1) Correction: a*s the sensor ability to measure the actual value close to the real.

The measurement error was proposed by [17], using the equation,

$$E = M - T \quad (1)$$

where E is the measurement error represents by the difference between the actual value T and the value measured by sensor M.

Thus, a physical sensor may have its accuracy value calculated by the equation,

$$C = 1 - \frac{E}{T} \quad (2)$$

where C is the correction value, E the error value and T, the real value. The correction is obtained by subtracting the relative error of 1.

2) Accuracy: the ability of a sensor to provide the same reading on the same measurement on equal terms. Unlike the correction which has a proximity to a true value, the accuracy shows the sensor proximity of successive readings, which can be represented by the equation:

$$P = \frac{true\ positives\ number}{total\ true\ positives\ and\ falses\ number} \quad (3)$$

where P is the accuracy value, true positives number represents the cases that have been correctly recognized as positives, and false positives those that have been incorrectly recognized as positive [16].

3) Time period: is the time interval between two measurements.

4) Sensor State: is related to the environment where the sensor is installed, and can be static (in the case of fixed sensors, such as temperature measurements) or dynamic, in case of integrated sensors on people or mobile device

5) Range: refers to the maximum distance that a sensor can collect a context measurement.

B. Analysis about Repository

The analysis of the repository is performed taking into consideration the sensors quality and the access level of its slices.

Regarding data access level we have the following:

- Public slice: in this type profile, all sensors arranged in this repository will be made available to any user in the system;
- Private slice: in the private slice, the sensor network manager does not allow access to users freely. In this way, only the users created by him or who have requested access by invitation will be able to access the data on these devices;
- Mixed slice: in this profile, the network manager can provide part of the sensors of your slice for any user. The other sensors will not be visible.

Regarding sensor quality we have the following:

The criteria selected to define the sensor quality will be based on correctness and accuracy, since they are objective data and not properly linked to the context.

The sensors will be classified by the sum of correction value with the accuracy value, by equation,

$$Qdt = C + P \quad (4)$$

where Qdt is the sensor quality content, C is the correction value and P the accuracy value.

The total quality of the repository sensors is calculated by equation:

$$QdtRep = \frac{sum\ of\ the\ indices\ for\ each\ sensor}{total\ sensors\ X\ 2} \quad (5)$$

 ISBN: 978-1-61208-438-1

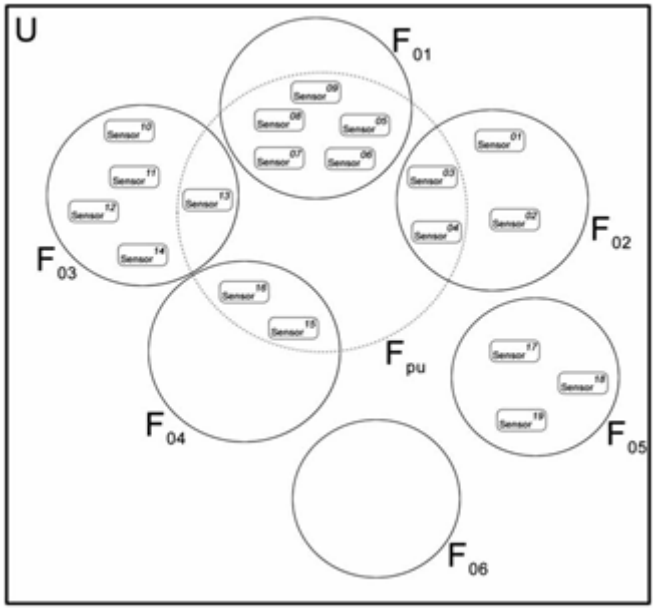


Figure 1. General representation of sensors repository

Below, we find the calculation to assess the repository where Fpu is the public slice and represents total public sensors, being responsible for delivering the information from the sensors contained in this slice for any system user, and U, which represents the total sensors contained in the repository. The universe of sensor system is the sum of all the slices, represented by equation,

$$\text{QftRep} = \frac{Total\ public\ sensors}{Total\ sensors} \quad (6)$$

where QftRep represents the quality of slices repository.

From these formulas, it is possible to set criteria in order to assess the repository based on the slices and the sensors characteristics, by equation:

$$Qrep = \frac{QdtRep + QftRep}{2} \quad (7)$$

Qrep represents the repository quality through the sums of the sensors qualities and the slices divided by 2. Thus, a higher quality repository is one in which Qrep is closer to 1.

In Figure 1, it is possible to observe the sensors and their respective slices together. Each manager would be responsible for one of the numbered slices. These could be exclusively private (F05), exclusively public (F01 and F04), mixed (F02, F03 and F04) or empty (F06).

For quality of the sensors, will be applied the following values, as shown in Table 1.

TABLE I. TABLE SENSOR CHARACTERISTICS

Item	Access level	Correction	Accuracy	Index
Sensor01	Private	0,3	0,4	0,7
Sensor02	Private	0,9	0,9	1,8
Sensor03	Public	0,5	0,8	1,3
Sensor04	Public	0,9	1,0	1,9
Sensor05	Public	0,5	0,5	1
Sensor06	Public	0,5	0,5	1
Sensor07	Public	0,8	0,8	1,6
Sensor08	Public	1,0	0,8	1,8
Sensor09	Public	1,0	1,0	2
Sensor10	Private	1,0	1,0	2
Sensor11	Private	0,5	0,9	1,4
Sensor12	Private	0,8	0,8	1,6
Sensor13	Public	0,8	0,8	1,6
Sensor14	Private	0,2	0,6	0,8
Sensor15	Public	0,9	0,9	1,8
Sensor16	Public	0,5	0,8	1,3
Sensor17	Private	1,0	1,0	2
Sensor18	Private	1,0	1,0	2
Sensor19	Private	1,0	1,0	2
Total				**29,6**

From this scenario, it is possible to assess this repository and the system can assign value to it.

Applying the formulas for the scenario presented in Figure 1, we have the following data in Table 2:

TABLE II. SCENARIO

Equations Aplication	Index
$\text{QdtRep} = \frac{29,6}{38}$	0,78
$\text{QftRep} = \frac{10}{19}$	0,53
$\text{Qrep} = \frac{0,78 + 0,53}{2}$	0,66

The closer to 1 value, the better the repository quality, from the characteristics of each sensor as the access level assigned to them by means of each slice.

C. Analysis about data search engine

The search engine is the service responsible for the search of sensors available for each user profile. This search is made in slices, where the user has access. By default, any user can receive research data from the primary slice. The search will retrieve data from other slices only if their available sensors are marked as public.

In the survey, the user can enter a sentence with the parameters that are related to desired data. For example, if he wants air humidity data in a particular city, the sentence could be "humidity Sao Paulo." The search interface then looks for the repository data based on this query.

As a proposal for a model of sensor channels, a representation of this channel was implemented, as the class in Figure 2. From this implementation, it is possible to define the search engines on their attributes.

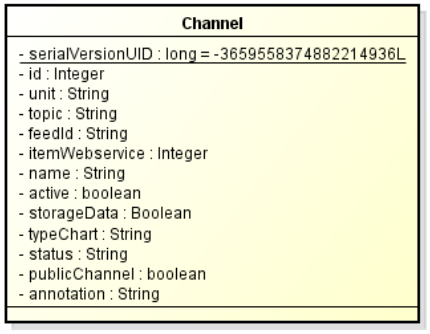


Figure 2. Sensor channel class

Search engines perform an analytical research on the unit, name and annotation attributes of the channel object.

Among the attributes of these objects, these three were chosen because they represent the measurement characteristics. The attribute unit refers to the measurement unit used by the channel, e.g., °C (degree Celsius) for temperature

The name field refers to a measurement identification name such as "TEMP". Finally, the annotation field serves as a comments field about this sensor channel, where the sensor manager could enter comments about it, as "Temperature capture in the Boa Vista neighborhood in Recife".

The analytic research is done by Hirbenate Search library implementation, which is a tool that integrates the Apache Lucene technology of complete search engine for text, with implementation by Index Hibernate by domain from notes, index database synchronization through objects [18].

The implementation of this library is made on the object channel by setting the unit, name and annotation attributes, with the definition of the Ngram type parameters, which is a feature of this API for data analysis, filtering search of words with 3 letters at least. This way, the rates analyzers can recover data even from typing error. In the option used by the application, if the word search is temperature, the analysis could be made to tem, emp, mpe, per, era, rat, atu, tur and ura.

Another definition was chosen so that, regardless of typing the whole search was made in lowercase. Thus, if the word is "Temperature" the system switches to "temperature" and will do the analytical search

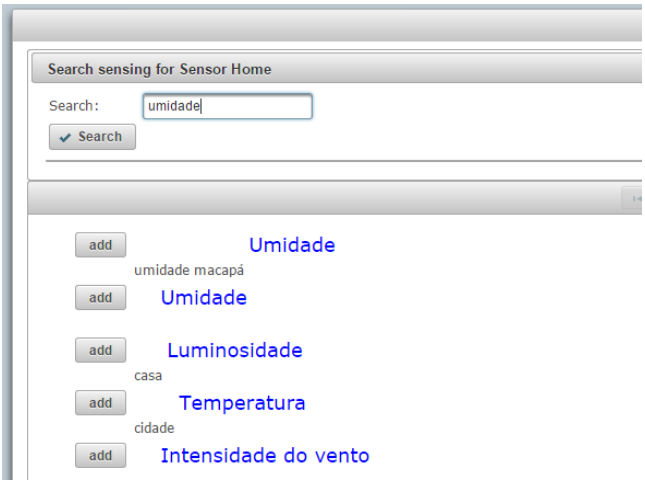


Figure 3. Sensor searcg screen

In Figure 3, it can check the implementation of the search functionality in use. In this example, the search word has "humidity". The search returns an objects list according to the relevance of what was searched. Thus, the first item in the list returned is a sensor named "humidity." As there were two registered, which has more fields of relevance is listed first. The first object contains both the word humidity in the name field, as in the annotation, while the second object only has this word in the name field.

The other results are listed by the analytical filter Ngram. The "Brightness" channel returned from the search "umi", "dad" and "ade", because this words combination is in the name field of this object, while the value "Temperature" was returned in the search because in its annotation field there is the word "city" that contains the Ngram, "dad" and "ade" atributes.

Therefore, with the library implementation, it is possible to perform the search of the measurement channels represented by the channel object in the database by analytical research into text.

D. Validation and Results

The final analysis and more important for class SaaS was sensors search. The parameter used to test was the response time and research relevance, as well as the data indexing time in the bank.

Three distinct databases were created for each scenario, to evaluate the sensor search performance, which is the main functionality for the end user, because it will serve as a data source for the sensors consumption as a service.

The scenarios were created as follows: scenario 1, with the amount of 9 sensors and 15 channels; scenario 2, with the amount of 99 908 sensors and 202 625 channels; and scenario 3, at the amount of 300,000 sensors and 3,000,000 sensor channels.

The search response time will serve directly to the end user because it will measure the time between the data request and the return of the time, which will influence the user experience relative to the solution.

The term "sensor recife" surveyed in Google returns a data set around 400,000 items, with the seek time of 650 milliseconds. This time was used as a reference to act on the user experience about extensively used service.

The time was classified into five levels, as follows:

- T1: simple search, with 1 word in 1 channel fields;
- T2: composed search, with 2 words in 1 channel fields;
- T3: composed search, with 2 words in 2 channel fields;
- T4: simple search, with 1 word with highest incidence in 1 channel fields;
- T5: composed search, with 2 words with highest incidence in 2 channel fields.

The search relevance analyses if the request returned to the user the data expected for him in the query.

The indexing time is a system parameter for assessing the time that the solution takes to index data in the database with the use of indexing through Hibernate Search. Table 3 shows the data obtained from the scenarios presented.

TABLE III. INDEXED QUERY SCENARIOS

Scenarios	T1	T2	T3	T4	T5
Scenario 1	8ms	15ms	16ms	11ms	16ms
Scenario 2	16ms	20ms	32ms	99ms	145ms
Scenario 3	21ms	28ms	41ms	139ms	283ms

The search time in three scenarios was well below the reference value for the proposal made for the solution. Thus, the user would receive your request, in the worst scenario, in 283ms.

 ISBN: 978-1-61208-438-1

The relevant factor is related to the quality of the data found with what the user was really looking for. Based on fuzzy feature, which delivers the term equal or similar to what was requested, it was asked to an users group a questionnaire that contained a description of the search task of sensing. Based on the results, the user would answer the questionnaire and then analyze it as shown in Table 4, with the following information, based on criteria listed below:

- Low: does not contain the requested data;
- Medium: contains the data similar to the requested;
- High: contains exactly the requested data.

TABLE IV. SEARCH RETURN RELEVANCE BY THE USER

Relevance level	P1	P2	P3
Low	0	0	0
Medium	0	1	0
High	7	6	7

Based on the data presented in the research referred to as P1, P2 and P3, the user found the requested data in the search.

The indexing time is a system parameter that influences indirectly in the sensors search. By means of this feature, the sensors are indexed and may be consumed by the search service more efficiently, through a text search. At the time, the solution is indexing only on its startup, however other strategies can be made so that this occurs at other times as well. The problem with indexing is that when it occurs, it consumes a lot of server performance, in addition to disable access to the base. One solution would be to perform indexing in minor peak times. However, as it is a cloud application and it can be accessed anywhere in the world, this time could not meet the solution, since the data flow in a particular location could be lower, but in another could be higher. The most recommended would be selective indexing, with the inclusion of new data as they were entering the base, but, over time, a full indexing would be required to keep the data as possible optimized on record.

TABLE V. SENSORS INDEX DATA

Scenarios	Total indexing time	Average indexed documents per second
Scenario 1	< 1ms	15
Scenario 2	57.025ms	3552,82
Scenario 3	280.234ms	3858,21

The indexing data were summarized in Table 5 with the following analysis:

- In scenario 1, the value was negligible, due to the low amount of data and, therefore, this value was expected
- Scenarios 2 and 3 showed indexing time values proportional to the size data in the database. Thus, the larger the database, the greater is the time for which such data to index.

These scenarios showed values close when considering the average of indexed documents per second. Such value is close because it is influenced by the server computing resources, as disk access time and processing. This value can be optimized by improving the cloud hardware.

IV. CONCLUSION

This paper presented a proposal for analysis in systems that work with sensors as a service, upon evaluation of the repository and data engine consumption. From this approach, it is possible to qualify systems that operate in this paradigm, setting a quality level and therefore provide the user with a better service. As future work other criteria can be added to assess the repository as characteristics related to performance, security, large amounts of data among others. This would increase the range of solutions that could be met from these analyses.

REFERENCES

[1] G. Press, "Internet of Things By The Numbers : Market Estimates And Forecasts", http://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/, 2014 [retrieved: 03, 2015].

[2] H. Sundmaeker and A. Saint-exupéry, De, "Vision and Challenges for Realising the Internet of Things", CERPIoT, Luxembourg, 2010.

[3] ABI Research, "More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020", https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne, [retrieved: 11, 2014].

[4] H. Schaffers, N. Komninos, and M. Pallot, "Smart Cities as Innovation Ecosystems Sustained by the Future Internet", Fireball White Paper, EU, 2012.

[5] M. F. Catela, C. D. Pedron, and B. A. Macedo, "Service level agreement em cloud computing: um estudo de caso em uma empresa portuguesa". Univ. Gestão e TI. 4, 2014.

[6] N. W. Khang, "CLOUD COMPUTING SOLUTIONS: IAAS, PAAS, SAAS", http://wptidbits.com/techies/cloud-computing-solutions-iaas-paas-saas/, [retrieved: 08, 2014].

[7] G. Aceto, A. Botta, W de Donato, and A. Pescapè, "Cloud monitoring: a survey", Comput. Networks. 57, 2013, pp. 2093–2115.

[8] C. A. Kamienski, D. F. H. Sadok, E. M. Azevedo, R. A. M. B. K. Simões, and S. F. de L. Fernandes, "Um Modelo Integrado de Composição em Nuvem Computacional Conteúdo", Recife 2011.

[9] J. Simão and L. Veiga, "A classification of middleware to support virtual machines adaptability in IaaS", Proc. 11th Int. Work. Adapt. Reflective Middlew. - ARM '12, 2012, pp. 1–6.

[10] F. R. C. Sousa, L. O. Moreira, and J. C. Machado, "Cloud Computing: Concepts, Technologies, Applications and Challenges ", Escola Regional de Computação Ceará - Maranhão - Piauí., Parnaíba, 2009, pp. 25.

 ISBN: 978-1-61208-438-1

[11] A. Botta, W. Donato, V. Persico, and A. Pescap, "On the Integration of Cloud Computing and Internet of Things", FiCloud. Barcelona, 2014, pp. 8.

[12] A. A. Gavlak, L. Muratori, and D. A. Graça, " Image as a Service (IaaS): satellite images digital processing ZY-3 via web ". XXVI Congresso Brasileiro de Cartografia e V Congresso Brasileiro de Geoprocessamento. Gramado-RS 2014.

[13] X. Sheng, X. Xiao, J. Tang, and G. Xue, "Sensing as a service: A cloud computing system for mobile phone sensing". 2012 IEEE Sensors, 2012, pp. 1–4.

[14] P. Banerjee, C. Bash, R. Friedrich, P. Goldsack, B. A. Huberman, and J. Manley, "Everything as a service: Powering the new information economy", Computer (Long. Beach. Calif). 44, 2011, pp. 36–43.

[15] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a Service Model for Smart Cities Supported by Internet of Things", Trans. Emerg. Telecommun. Technol. 2013, pp. 1–12.

[16] A. Manzoor, "Quality of Context in Pervasive Systems : Models, Techniques and Applications" Tu, Wien. 2010, pp. 155.

[17] J. G. Webster, "The measurement, instrumentation, and sensors handbook", CRC Press, 1999.

[18] JBoss, "Hibernate Search", http://docs.jboss.org/hibernate/search/3.4/reference/en-US/html_single/#preface, [retrieved: 01, 2015].

 ISBN: 978-1-61208-438-1

Middleware Applied to Digital Preservation: A Literature Review

Eriko Brito, Paulo Cesar Abrantes, Bruno de Freitas Barros
Recife Center for Advanced Studies and Systems (CESAR)
Recife – PE, Brazil
E-mails: eriko.brito@outlook.com, {pc.abrantes, barrosbruno}@gmail.com

Abstract — **Maintaining digital collections available for humanity use at long term is a big challenge for the areas of digital preservation, management policies of curator centers and technologies for data reproducibility. This paper performs a literature review to investigate middleware options for digital preservation, listing its main features and applications. Seven solutions were found and it was concluded that the cataloged technological bases are mature enough, which indicates an optimistic future for the digital curation area.**

Keywords—Digital Curation; Digital Preservation; Reproducibility; Middleware.

I. INTRODUCTION

Research and solutions in the digital preservation area have evolved significantly in recent decades establishing their technological, methodological and political apparatus. Brito et al. [1] point out that compared to the physical collections preservation, digital content brings an association, almost paradoxically, between a great potential risk and a great potential for protection. The potential risk is represented by the ephemerality of digital storage that can be irretrievably lost because of technical or human failure much more easily and quickly than in the case of physical representations of content. The potential for protection, in turn, is anchored in the fact that digital collections can be endlessly reproduced and stored with full fidelity and integrity.

The continuity of digital collections depends, mostly, on implementing strategies that take full advantage of the potential for protection, attempting to neutralize its inherent potential risk. However, the challenge can represent much more of a social and institutional problem than a purely technical issue, because, particularly concerning digital preservation, it depends on institutions that undergo changes of direction, mission, administration and funding sources, as Arellano defends [2].

Concomitantly, in the information technology area, distributed systems are established as an information-sharing pattern. That leads us to cloud computing that, according to Mell et al. [3], is a ubiquitous, convenient and on demand model for sharing computing resources that can be managed and made available with minimal effort.

Applying the distributed processing principles and cloud computing to the maturity scenario of digital preservation seems to be an only natural option. Wittek et al. [4] relates distributed systems to distributed digital preservation, pointing out that it can ensure the replication of digital artifact copies between geographically separated servers. It is important to say that distributed digital preservation is not only the act of ensuring the backup of digital artifacts, but also the possibility to access the data over the years and to reuse it.

Among the distributed systems and the digital preservation, there is the middleware, which, for Rocha et al. [5], is the group of components located between the operating system and the application, promoting generic services to support the execution of distributed applications. For this Literature Review (LR), the presented concept of middleware is extended to a layer situated between the business rules of the curator center and the supporting IT infrastructure of digital preservation.

The remainder of the paper is structured as follows. Section 2 presents the review planning with its goals. In Section 3, an overview of the middleware options found is presented. Section 4 will show a detailed analysis of each option and, finally, Section 5 presents conclusion and register for future work.

II. REVIEW PLANNING

This review follows the guidance of Kitchenham et al. [6] in its structure. It contains the objectives of the review, the research questions to be presented, the criteria used for the negative scope of the research, the strategy that will be used and the way it will be conducted. It is expected that in this way, other researchers can repeat the procedure according to their own definitions.

A. Objectives

The goal of this LR is to identify existing options in literature of middleware solutions used in digital preservation processes of curation centers. The result of this work can provide the discovery of challenges and trends in this research field.

B. Research Question

The research question that guides this LR is: what middleware options for digital preservation are currently available in the literature?

C. Exclusion Criteria

For the established research question, it was decided that some productions will be excluded from the scope of this LR. Specifically, scientific productions that:

- Are in proposal stage;
- Present the state of art for the research question;
- Were published before the year of 2010.

 ISBN: 978-1-61208-438-1

D. Research Strategy

The research strategy consists in establishing the premises that will be considered by the LR to achieve its goals. Thus, in this section, will be elicited the sources of the research project, base language and keywords to be used in the search engines listed at the sources. These assumptions are defined as follows:

Sources: IEEE Xplorer, Science Direct, ACM Digital Library, Compendex and Scopus.
Language: the English language will be used as reference for the LR, as it is considered the most popular in the scientific world.
Keywords: Middleware to digital preservation, Distributed systems for digital preservation, Electronic Records Archives capabilities, reproducibility.

E. Review Conduction

This paper was planned and produced in May 2015 in response to the approval requirements of the discipline of Systems Interoperability of the Professional Master's degree in Software Engineering at CESAR.EDU. The sources were found by using search strings formed by logical combinations of keywords presented as follows:

("digital *preservation"* OR "digital *curation"* OR *reproducibility* OR *cloud* OR "*distributed systems*") AND *middleware*

The collection of articles was obtained by reading the abstracts, guided by the research question and exclusion criteria presented. As a result of this approach, there were seven relevant papers to the theme of this LR, which are presented in Section 3.

In summary, the results obtained from the data sources were:

a) IEEE Xplorer returned 37 works, out of which 3 were considered aligned to the research question.
b) ACM Digital Library returned 47 results, some of them also found in IEEE Xplorer, and 4 of them were more relevant to this LR.
c) In Science Direct 20 results were located.
d) In Scopus, 7 and,
e) In Compendex, 5 results.

Results **c**, **d** and **e** were classified as outside of the LR scope either because they match the exclusion criteria or because the abstracts were not aligned with the research question.

III. Overview of Selected Options

This section is dedicated to present the summarized results that are the most relevant to this LR. The following seven subsections are identified by the titles of the papers. They describe relevant aspects of each result and its purpose.

A. Digital Preservation in Grids and Clouds: A Middleware Approach

Digital preservation can be seen as an effort to retain, as long as necessary, digital material for future use on research, consultations or any other form of knowledge management.

Several of the current digital preservation systems are backed in the computational grid technology, but the advent of cloud computing and its potential has become a strong and attractive possibility [4].

Placed in the business layer of the SHAMAN model, the proposal made by Peter Wittek and Sandor Daranyi suggests a middleware which is flexible enough to enable a quick and transparent switching between cloud computing and grid computing, in compliance with business rules and requirements of the entity that needs to preserve its digital collection [7]. Figure 1 shows the proposed architecture that includes, on the left, an archive layer governed by a set of pre-established policies and, on the right, computational scalability components in clouds or grids.

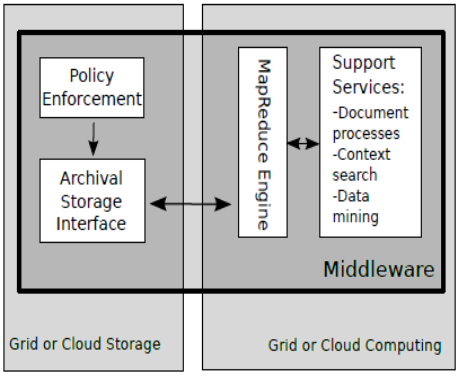


Figure 1. Overview of the Peter and Sándor's proposal

The considerations of the authors suggest that small businesses can be the biggest beneficiaries of the switching flexibility, by replacing servers or grid acquisitions with service level agreements with computing service providers in the clouds.

B. Content server system architecture for providing differentiated levels of service in a Digital Preservation Cloud

Quyen L. Nguyen and Alla Lake [8] write about storage challenges and resulting preservation of the rapidly growing volume of digital records and the need for some companies and industries to stick to directives such as Sarbanes-Oxley Act. It is important to note that digital preservation covers the need of keeping information available and accessible regardless of the hardware and features that originated it.

In this context, and to the authors, preservation in the clouds is a simple and economical option for models such as the Open Archive Information System (OAIS), as seen in Figure 2, which requires great engineering effort and planning.

 ISBN: 978-1-61208-438-1

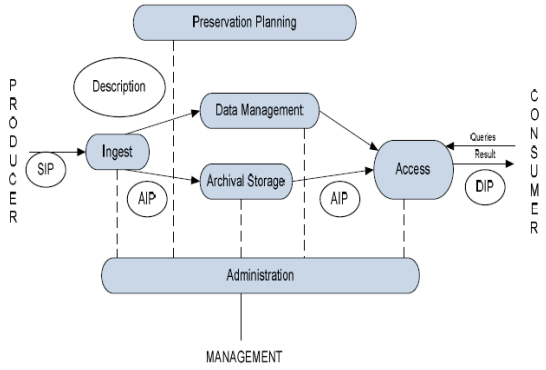


Figure 2. The OAIS Model

In the LDPaaS proposal, the Ingest layers, Preservation, Archival Storage and Access are abstracted and offered as individual services in the clouds. With this arrangement the proposal provides differentiated service levels for the various needs of long term digital preservation.

C. *Biopolis, long term preservation of digital user content*

The purpose of the Biopolis is that the digital contents of its users can be replicated in the clouds, commercially or not. With regard to copyright, the design ensures the ultimate relationship between the author and the maintainer of the digital material.

Differently from systems like Google Panoramio and Flirk, in which, as Sardis et al. [10] state, time stamps are not present, the Biopolis project supports time attribute and is prepared to offer, in the coming years, scalable storage, preservation and organization through libraries and semantic searches if needed, stamping data with its registration time.

Through the Internet, the users can add their digital content via the web interface of the Biopolis system, by logging the geographic position of upload and copyrights for appropriate action. After the content for retention time setting is filtered, storage, procurement, distribution, preservation, recovery and reuse options are provided.

In terms of middleware, the Biopolis project has levels with its own API functions, which are used as clients in a common runtime environment providing scalability, high availability and routing messages. Figure 3 shows the middleware components.

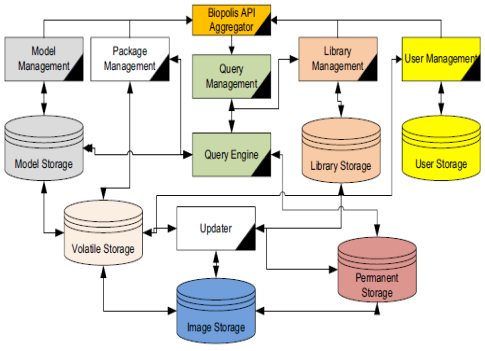


Figure 3. Biopolis Middleware

D. *PDS cloud: Long term digital preservation in the cloud*

The Preservation DataStores (PDS) proposal is a preservation cloud based on the OAIS model, developed by Ccsds [9] as an infrastructure component of the European Union ENSURE. It employs multiple heterogeneous providers and embodies the concept of object-informational preservation.

The authors conducted a gap analysis concluding that "just throwing" the data in a cloud is not the solution for preservation repositories. Instead, a more professional approach is expected. The main features of PDS Cloud includes: a) multiple clouds storage support, b) enhancement of future understandability of content by supporting data access using cloud based virtual appliances and, c) advanced services based on the OAIS model.

As shown in Figure 4, the PDS Cloud architecture is implemented as a middleware, composed by a broker that OAIS interconnects between multiple entities and the cloud. On the front-end, PDS Cloud exposes to the client a set of OAIS-based preservation services such as ingest, access, delete and preservation actions on OAIS Archival Information Packages (AIPs).

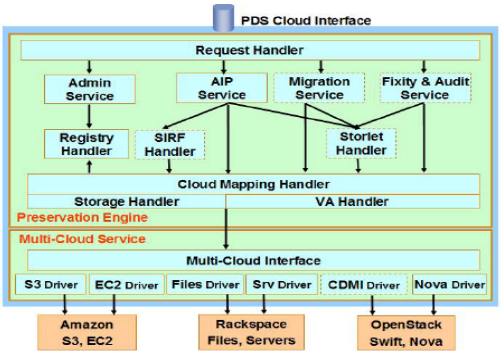


Figure 4. PDS Cloud Architecture

On the back-end, it leverages heterogeneous storage and computes cloud platforms from different vendors. AIPs may be stored on multiple clouds simultaneously to exploit different storage cloud capabilities and pricing structures, and to increase data survivability.

In the conclusions, Rabinovici-Cohen et al. [11] point out that the main purpose of PDS Cloud is to keep the responsiveness of long term digital material by adhering to the changes of technological scenarios.

E. *Rule-based curation and preservation of data: A data grid approach using iRODS*

In this paper, Hedges et al. [12] present the implementation of a data management layer to support a system of preservation research data. For data storage, the authors suggest the use of the e-Science technology and grid computing middleware, presenting how integrated Rule-Oriented Data Management System (iRODS) can be used to implement complex strategies of digital preservation.

The contextualization involves the consideration that the Storage Resource Broker (SRB) developed by the San Diego

 ISBN: 978-1-61208-438-1

Supercomputer Center (SDSC) is the most widely used data management middleware for digital preservation. The iRODS, open source, is presented as its successor, also developed by SDSC, with significant evolution especially in the political representation of capacity in terms of rules.

This feature of the iRODS middleware allowed the authors to explore two key points:

- preservation actions taken when a digital resource is ingested into an iRODS data grid;
- post-ingest management of the integrity and authenticity of curated digital resources.

In its conclusion, the paper exalted the iRODS skills, such as the flexibility to implement the rules in a sequence of actions to be executed in particular contexts or when certain events like the ingest of the file into the grid, or a timer occur.

F. New Roles for New Times: Digital Curation for Preservation

The work of the authors was to examine tools and techniques used to automate the exchange of significant data volumes between MetaArchieve Cooperative, which uses "Lots of Copies Keep Stuff Safe" (LOCKSS) and the Chronopolis preservation system, which uses the Storage Resource Broker (SRB). It is expected that this work enable the use of preservation systems to share data between these two preservation networks in the United States of America [13].

Staff from the MetaArchive and Chronopolis are investigating technologies that allow data exchange between LOCKSS and iRODS, based on real-world, practical implementations within MetaArchive, Chronopolis and CDL. Three methods of exchange are being developed and evaluated.

The first method uses BagIt and related technologies to package and transfer large collections efficiently and reliably. A second more, sophisticated tool, allows the user to identify an existing BagIt bag, transfer its contents, ingest these objects into a storage zone and perform quality assurance testing for the whole process. The third proposed method utilizes a LOCKSS plugin. LOCKSS has its own technical architecture that can be enhanced by the use of custom-created, XML-based plugins, which allows data to be manipulated according to defined rules. They will create plugins that will allow the LOCKSS system to interact with an iRODS system.

In its conclusion, the paper emphasizes the importance of the project that will integrate two major digital data preservation platforms and, by doing so, improve multi-disciplinary and multi-institutional scientific exploration that is highly data-driven. An integrated LOCKSS/iRODS infrastructure will better support the growing diversity in formats, visualization, and analytical tools that empower researchers to utilize information and data more effectively.

G. Semantic Middleware for E-science Knowledge Spaces

Futrelle et al. [14] present in his paper a middleware called Tupelo, which implements Knowledge Spaces. It enables scientists to find, use, relate and discuss data and metadata work in a distributed environment. Its construction is based on a combination of semantic web technology for data management and workflow. The main benefit of Tupelo middleware is the simplification of interoperability by providing the Knowledge Space view of heterogeneous resources distributed in institutional repositories.

In architecture terms, Tupelo is based on an abstraction called "context", which represents a kind of semantic view of distributed resources. Context implementations are responsible for performing as "operators", which are atomic descriptions of requests to either retrieve or modify the contents of a context. Two primary kinds of operations are provided:

1. Metadata operations, including asserting and retracting statements (i.e., RDF statements) and searching for statements that match a query; and
2. Data operations, including reading, writing, and deleting binary large objects (BLOB's), each of which is identified with a URI.

In its closing remarks, the paper suggests that Tupelo's interoperability-based architecture allows it to be used to connect, without replacing or displacing, existing software stacks to add context and help integrate the heterogeneous aspects of large-scale scientific work, including observation, analysis, organization, and publication.

Another significant consideration was related to reducing the development effort required to support scientific domains, allowing an active view of scientific work with strong guarantee of reusability, based on explicit semantics and declarative descriptions of analytic processes, opening new opportunities for more effectively disseminating and preserving the fruits of ongoing, evolving scientific discovery.

IV. CONCLUSION

This paper presented a literature review of middleware available for the area of digital preservation. The main objective was to list options available for this context and to present their main characteristics and applicability.

It was possible to observe the complexity of the long-term digital preservation process, and that this is still an evolving model. It was contextualized the term middleware as a layer placed between the business rules of the curator center and the supporting IT infrastructure of digital preservation.

Thus, it was possible to map the following results as the most cited middleware related to digital curation and preservation:

a) The Storage Resource Broker (SRB), developed by the San Diego Supercomputer Center (SDSC);

 ISBN: 978-1-61208-438-1

b) iRODS, open-source, presented as its successor and also developed by SDSC, and the;
c) LOCKSS Program as an open-source, library-led digital preservation system built on the principle that "lots of copies keep stuff safe".

Table 1 shows an overview of the solutions found, being organized by the name of the paper, the model used as the base of the solution, the use of grids or clouds and its main characteristics.

TABLE 1. SOLUTIONS OVERVIEW

Paper Title	Base model	Grid	Cloud	Main Characteristic
Digital Preservation in Grids and Clouds: A Middleware Approach	SHAMAN	Yes	Yes	Allows a transparent switching between cloud computing and grid computing
Content server system architecture for providing differentiated levels of service in a Digital Preservation Cloud	OAIS	No	Yes	Abstracts the layers and offers them as individual services in the clouds.
Biopolis, long term preservation of digital user content	None	No	Yes	Provides scalability, high availability and routing messages.
PDS cloud: Long term digital preservation in the cloud	OAIS	No	Yes	Supports multiple clouds storage and provides data access using cloud based virtual appliances
Rule-based curation and preservation of data: A data grid approach using iRODS	SRB	Yes	No	Implements a rule-oriented data management layer to support a system of preservation research data
New Roles for New Times: Digital Curation for Preservation	LOCKSS and iRODS	Yes	Yes	Allows data exchange between LOCKSS system and iRODS
Semantic Middleware for E-science Knowledge Spaces	None	Yes	No	Simplifies the interoperability by providing the Knowledge Space view of heterogeneous resources distributed in institutional repositories

Although Brito et al. [1] claim that the digital preservation area is still in the early stages of its formation and that the technological, methodological and political apparatus to preserve digital information is still being built, it was found mature middleware options related to digital preservation and access to long term information.

After this research, it was realized that information security applied to digital curation is an area that can be explored, so as future work is suggested a review of the literature to list the existing solutions.

ACKNOWLEDGMENT

To the CESAR.EDU teaching staff that contributed with methodological guidance for the development of this paper. We also appreciate the patience and dedication of our families that unconditionally supported the work that has been done so far.

REFERENCES

[1] E. Brito, R. Costa, A. Duarte, P. De Pós-graduação, and I. Ppgi, "The adoption of model canvas in data management plans for digital curation in research projects," 2012.

[2] M. Arellano, "Digital Preservation Criteria of Scientific Information," Brazil: University of Brasilia, 2008, p. 50.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology," Natl. Inst. Stand. Technol. Inf. Technol. Lab., vol. 145, 2011, p. 7.

[4] P. Wittek and S. Darányi, "Digital Preservation in Grids and Clouds: A Middleware Approach," J. Grid Comput., vol. 10, no. 1, 2012, pp. 133–149.

[5] V. H. Rocha, F. S. Ferraz, H. N. De Souza, and C. A. G. Ferraz, "ME-DiTV : A middleware extension for digital TV," International Conference on Software Engineering Advances, 2012, pp. 673-677.

[6] B. Kitchenham, et al., "Systematic literature reviews in software engineering – A tertiary study," Inf. Softw. Technol., vol. 52, no. 8, Aug. 2010, pp. 792–805.

[7] P. Innocenti, et al., "Assessing digital preservation frameworks: the approach of the SHAMAN project,", 2009, pp. 412–416.

[8] Q. L. Nguyen and A. Lake, "Content server system architecture for providing differentiated levels of service in a Digital Preservation Cloud," Proc. IEEE 4th Int. Conf. Cloud Computing, 2011, pp. 557–564.

[9] Ccsds, "Reference Model for an Open Archival Information System (OAIS)," Forsp. Data Syst., no. January, 2002, pp. 1–148.

[10] E. Sardis, A. Doulamis, V. Anagnostopoulos, and T. Varvarigou, "Biopolis, long term preservation of digital user content," IEEE 10th Int. Conf. on e-Business Engineering, Sept. 2013, pp. 478-483.

[11] S. Rabinovici-Cohen, J. Marberg, K. Nagin, and D. Pease, "PDS cloud: Long term digital preservation in the cloud," Proc. IEEE Int. Conf. Cloud Eng. IC2E, 2013, pp. 38–45.

[12] M. Hedges, A. Hasan, and T. Blanke, "Management and preservation of research data with iRODS," in Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience - CIMS '07, 2007, p. 17.

[13] T. Walters and K. Skinner, "New Roles for New Times: Digital Curation for Preservation," Washington: Association of Research Libraries, 2011.

[14] J. Futrelle, et. al., "Semantic middleware for e-Science knowledge spaces," Concurr. Comput. Pract. Exp., vol. 23, no. 17, 2011, pp.2107-2117

 ISBN: 978-1-61208-438-1

Middleware For Heterogeneous Healthcare Data Exchange: A Survey

Carlos Andrew Costa Bezerra [1 2] , André Magno Costa de Araujo [1 3], Bruno Sacramento Rocha [2], Vagner Barros Pereira [2], Felipe Silva Ferraz [2 3]

ITPAC – President Antonio Carlos Tocantinense Institute Araguaina, Brazil [1]; CESAR – Recife Center for Advanced Studies and Systems Recife, Brazil [2]; Federal University of Pernambuco Recife, Brazil [3]

e-mail: andrew@r2asistemas.com.br, amcaraujo@gmail.com, bdsr74@gmail.com, vagnerbarrospereira@gmail.com, fsf@cesar.org.br, amca@cin.ufpe.br, fsf3@cin.ufpe.br

Abstract — In this paper, we present a survey for data exchange middleware in health based on the HL7 standard. HL7 is an international standard, grounded on the Open System Intercommunication model (OSI), which standardizes exchange and transportation of information between healthcare organizations. Based on this standard, we examine examples of middleware selected during an exploratory research in the repositories of the Association for Computing Machinery (ACM) and Institute of Electrical and Electronics Engineers (IEEE). This article provides an overview of features present in the selected middleware.

Keywords - Middleware; Healthcare; Data Exchange; Interop; Heterogeneous Data; HL7; EHR;

I. INTRODUCTION

Healthcare systems produce a huge amount of health data. In most cases, information is spread among public and private institutions, clinics, care centers, laboratories, etc. One of the big challenges in the medical area is to provide access to patient's clinical data, regardless of where they were generated [1]–[3]. There are architectural models which standardize storage and communication of electronic healthcare records (EHR) – e.g. OpenEHR [4] and EN 13606 [5][6]. However, even solutions that have their working repositories implemented based on one of the previously mentioned models use different types of database or technologies, are in different versions or have other particularities that make them different from the others [7].

In the data exchange domain, there are various technologies that address the task of exchanging information between heterogeneous databases, such as Web-Services [8] and Cloud Services [9]. However, according to Xianyong Liu [10], middleware represents a technology that takes charge of communication and bridges lower-level Data Transfer Units.

For this paper, we discuss the already available HL7-based middleware, ongoing research projects and solutions under development for HL7-based healthcare information exchange. We conducted a literature review and selected two works which and will be henceforth named Middleware A and B. The other middleware analyzed represent solutions that are present on the IT market, which are Mitre hData, Mirth Connect and IBM Websphere [11]–[13]

Health Level-7 (HL7) refers to a set of international standards to transfer clinical and administrative data between software applications used by various healthcare providers. These standards focus on the application layer, which is "layer 7" in the OSI model [14][15]

The main objective of this work is to discuss the already available HL7-based middleware, as well as ongoing research projects and solutions under development for HL7-based healthcare information exchange. Section II discusses some aspects related to HL7 and healthcare systems. Section III presents the types of middleware and indications of use for each one. Section IV discusses projects and applications of HL7-based middleware and shows a comparison between them. Finally, conclusions and future developments of this research are presented in Section V.

II. HL7 AND HEALTHCARE SYSTEMS INTEGRATION

HL7 was founded in 1987 with the objective of defining standards for information exchange between health systems. It is a non-profit institution approved in 1994 by the American National Standards Institute (ANSI) [16]. The standard follows the conceptual definition of the application interface model, represented by the seventh layer of the OSI model, providing support to plug-and-play functionalities when used to integrate two or more health systems [17].

The HL7 standard provides specifications aiming to standardize the exchange and transportation of information between health systems, with the objective of making such systems interoperable [18]. HL7 is highly recommended to organizations seeking interoperability between internal and external systems of public health. The standard provides quality, efficiency and effectiveness in the delivery and sharing of medical information [17].

The HL7 standard encompasses groups of specification, like the ones exemplified below:

- Message protocols for the information exchange between health systems;
- Clinical Document Architecture (CDA) for document exchange.

A. Tools that use the HL7 standard

HL7 can be used for diverse implementations of health systems integration. Some of them were identified and described as follows:

- Implementation of test middleware to validate the formats of messages in the HL7 standard, allowing for the identification of a system that does not comply with the HL7 standard [19];

 ISBN: 978-1-61208-438-1

- Implementation of an emergency medical system using the Open Services Gateway Initiative (OSGi) service platform, enabling sharing information between medical systems and emergency services. This system uses the HL7 standard to format messages;
- Implementation of a mobile app that receives data from medical sensors and converts them to the HL7 standard so they can be sent later through the Wireless Local Area Network (WLAN) or Bluetooth. It can receive diagnoses and prescriptions later;
- Implementation of middleware to standardize the communication between different types of medical devices and health systems. It is made possible by converting the IEEE 1073 standard, when the medical devices follow the HL7 standard [17];
- Application that receives data from devices that measure vital signs, such as blood pressure, heartbeats, levels of glucose and body temperature. These pieces of data are received by the communication interfaces of the device, i.e. Universal Serial Bus (USB), Bluetooth or WLAN. All data is received by the application in different formats and then sent to the telemedicine central in the HL7 standard to be analyzed. When it becomes possible to provide more detailed care, the central activates a videoconference to detail and explain the situation to the caretakers in charge [20].

III. MIDDLEWARE

As a way to solve issues related to health information exchange, we analyze the middleware technology. It is a technology for distributed applications, able to hide details of the network and deal with a great amount of important functionalities for development, implantation, execution and interaction of applications [21]. The main idea is to be between two layers enabling communication between the connected parts. It is not only a network application which connects two sides, but also a means to promote the interoperability between the applications, protecting implementation details of functionalities and providing a set of interfaces to the customer [10].

There are various types of middleware that can be implemented with the objective of providing the exchange of health information, e.g. transactional, procedural, message-oriented and object-oriented.

A. Transactional Middleware

The transactional middleware (TM) is designed to provide synchronous distributed transactions [22]. It consists of a transactional monitor that coordinates simultaneous transactions between customers and servers, reducing the overload, response time and Central Processing Unit (CPU) costs between the components by guaranteeing the Atomicity, Consistency, Isolation and Durability properties (ACID). However, they offer unnecessary and undesirable warranties through the ACID properties. When a customer is performing a long processing, it may prevent other customers from following up with their requests [22].

As there is always the need to exchange a great number of health messages, each customer will always have a great amount of information to be shared, rendering this middleware inappropriate for this objective.

B. Procedural Middleware

The procedural middleware (PM) was developed by SUN Microsystems around the decade of 1980 and became known as the Remote Procedure Call (RPC). Its implementation is supported by various computing environments; however, they do not offer good communication because they do not support asynchronous transmission, replication or load balancing [22].

C. Message-Oriented Middleware

Message-oriented middleware (MOM) allows for distributed applications to communicate and exchange information by sending and receiving messages [23]. The essential elements for a message-oriented middleware are clients, messages and the MOM provider, as seen in Figure 1, which depicts an API and administration tools [23]. This middleware exchanges information asynchronously.

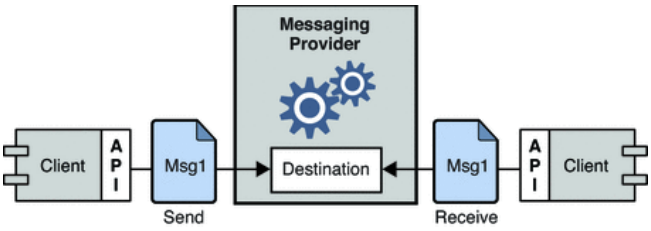


Figure 1. MOM-Based System [23]

This way of working provides the customer with the means to continue working even if a message has been sent – it doesn't make customer wait for a response. However, it might run out of message storage resources, which can generate a failure in the component. The HL7 international protocol predicts the exchange of information through asynchronous messages, which makes this type of middleware the best for implementing a solution for the health information exchange. Some examples that may illustrate this modality are Microsoft Message Queuing (MSMQ) by Microsoft, [24] and MQSeries by IBM [25].

D. Object-Oriented Middleware

The object-oriented middleware (OOM) allows applications to communicate and exchange information by invoking methods [26]. It works precisely like the local method invocation. Its communication is synchronous, which means an object invokes a middleware method and awaits the response of said method. Similarly to Peer-to-Peer and Remote Procedure Call (RPC) communications, this

 ISBN: 978-1-61208-438-1

workflow prevents the client from using the system while the invoked object is working.

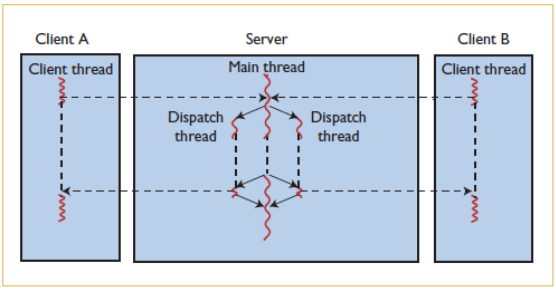


Figure 2. Synchronous call [26]

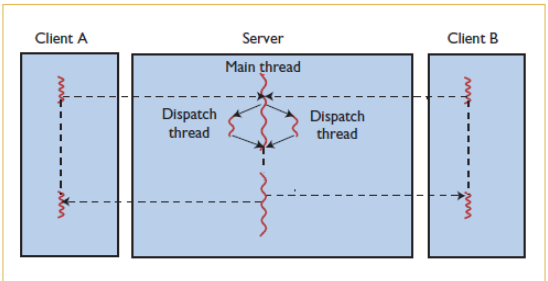


Figure 3. Asynchronous call [26]

A great advantage of the object-oriented middleware is the fact that it accepts the invocation of methods written in different programming languages [26]. Moreover, some OOM support both synchronous and asynchronous messages; this is an attempt to mitigate the limitations caused by one form of communication. Other OOMs also implement types of message exchange control in order to use resources more efficiently, by making use of threads and timeouts [26]. Figures 2 and 3 illustrate this behavioral change in Messaging.

IV. HL7-Based Middleware Analysis

The middleware analysis based on characteristics present in the construction of each middleware was separated into ten categories:

- Synchronous: when the middleware supports synchronous messaging;
- Asynchronous: when it supports asynchronous messaging;
- Type of Middleware: if the middleware is TM, PM, OOM or MOO;
- Web Bases: when the middleware runs uses web architecture;
- HL7 v2.x: when it supports any version 2 of HL7;
- HL7 v3.x: when it supports any version of HL7;
- HFIR: when it supports HL7 HFIR versions that combine the best features on v2, v3 and CDA;
- Parsing: when it offers the process to message syntactic analysis;
- Validating: when it offers the process of verifying the message conformance;
- Transmitting: when it offers the process of submitting the message to other client.

A. Middleware A

In [16], Liu et al. proposed an extensible HL7-based middleware, as shown in Figure 4, to provide a communication channel between different healthcare information systems that either did not support HL7 messages exchange or had not implemented an interface for it yet.

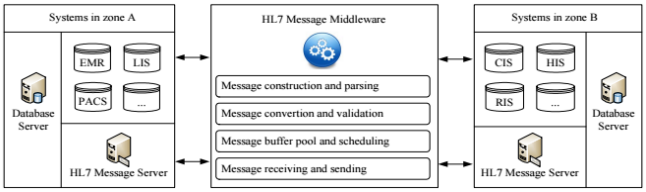


Figure 4. HL7 middleware architecture

This middleware has three deployment options: client-side, server-side and independent deployment., although only the independent deployment uses all the HL7 functions.

B. Middleware B

Ko et al. [27] presents a middleware framework developed for the National Taiwan University Hospital (NTUH). Figure 5 shows that it consists in a multi-tier service oriented architecture (SOA). It uses the message and event type to identify the required service and invoke the correspondent sub-routine. HL7 messages are used to format all information exchanged across systems.

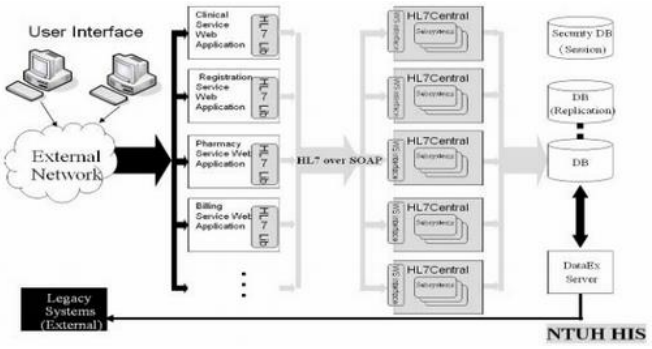


Figure 5. NTUH HIS system architecture Note

Besides the conventional middleware presented above, there are extensible libraries and components that address the complexities of HL7 encoding and decoding rules along with acknowledgements, allowing for the application developer to focus on underlying business logic and workflow, such as Merge HL7 and HAPI-Fast Healthcare Interoperability

 ISBN: 978-1-61208-438-1

Resources (FHIR) toolkits. Furthermore, there are specific tools for parsing and transmitting HL7 messages, e.g. HL7Spy and HL7 Analyst [28]–[30].

C. Mitre hData

Donald W. Simborg created the Level 7 protocol originally (in 1977), which later turned into the well-known HL7 standard version 2. He was developing departmental systems at the Johns Hopkins University, in Baltimore, MA, and was programming in the APL language. He continued to develop systems at the University of California in San Francisco (UCSF) in 1976, where his CIO was. Besides, he also worked for Mitre Corporation, a think-tank company of California where he acquired a lot of experience in the pioneer use of LANs and High Level Protocols (HLP) protocols [13].

hData is a standard for electronic health data exchange which is WEB-based and very light. Formulated in 2009 by MITRE and a non-profit organization, it has since evolved through the cooperation of the leaders in the health industry. The hData standard is the first one developed for RESTful health data exchange. Figure 6 explains the composition of hData.

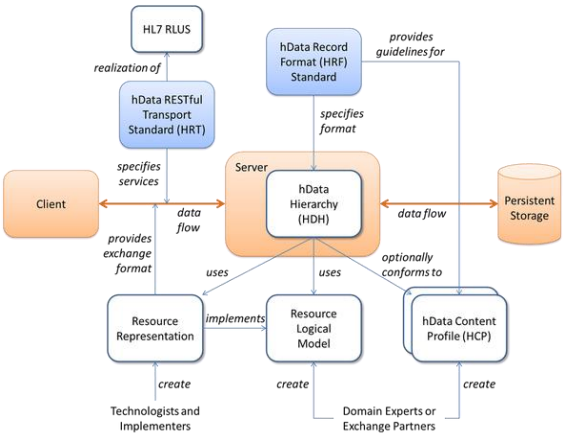


Figure 6. Relationship between the different components of hData [13]

The hData Registry Format (HRF) specifies the format of the hData Hierarchy (HDH). HDH is based on models of logical resources provided externally by experts in the domain or partners in the exchange of information, e.g. FHIR. HDH may be satisfied with one or more hData Content Profiles (HCPS), which are guides for the implementation and use of hData in specific domains. The server that hosts the HDH instance provides a service interface for the customers to interact with the HDH resources, placing information in persistent data storage. The hData RESTful Transport (HRT) standard specifies these services if there is a REST implementation.

D. Mirth Connect

Created by Mathias LIN, it is a middleware considered the Swiss army knife of the integration engines of health information. It is specialized and designed to exchange messages in the HL7 standard and counts on tools to develop, test and monitor interfaces [31]. Figure 7 illustrates an architectural vision of a system that uses Mirth Connect.

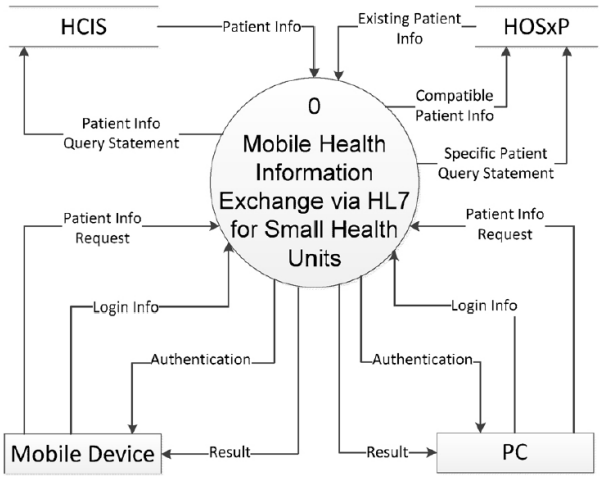


Figure 7. Architecture of a system using Mirth Connect as a Middleware [11] .

The following functionalities are available:

- A rich interface channel development and monitoring environment using an intuitive drag-and-drop template-based editor;
- Real-time connection monitoring through a dashboard;
- Message reprocessing;
- An integration server that supports a variety of protocols to connect to external systems, and diverse database options.

E. IMB WebSphere Message Broker

The IMB WebSphere Message Broker (WMB) is considered a product of business integration and used to integrate applications of general purpose by applying message transformation, enrichment and routing. It supports health applications even if they were built in other languages, such as .NET, C or Java. Figure 8 depicts the architecture of a WMB application.

WMB offers:

- Models of message used to analyze, route and transform HL7 messages;
- Input and Output nodes to integrate HL7 clinical applications;
- Integration with medical devices;
- Integration with Digital Imaging and Communications in Medicine (DICOM);
- Specific standards of medical assistance;
- Operational monitoring of data transmission for medical applications;
- Generation of events of Audit Trail and Node Authentication (ATNA) auditing to support

 ISBN: 978-1-61208-438-1

confidentiality of patient information, data integrity and provision;

- Ability to extract information from medical assistance data in message flows, and sending information.

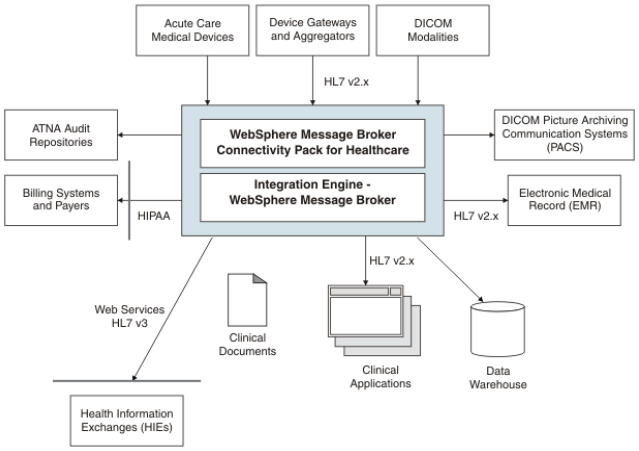


Figure 8. Architecture of a health system using WMB [12]

The selected characteristics of the presented middleware are summarized in Table I.

TABLE I. CHARACTERISTICS OF PRESENTED MIDDLEWARE

Characteristics	Mid A	Mid B	Websphere	Mirth Connect	hData
Synchronous	x	x	x	x	-
Asynchronous	-	-	x	x	x
Middleware type	MOM	MOM	MOM	MOM	MOM
Web-based	-	x	x	x	x
HL7 v2.x	x	x	x	x	-
HL7 v3.x	x	-	x	x	-
HFIR	-	-	x	x	x
Parsing	x	x	x	x	x
Validating	-	x	x	x	x
Transmitting	x	x	x	x	x

Table I shows the features supported by each middleware evaluated in this work. Observing the results obtained, it is possible to highlight the following points: i) all middleware analyzed are MOM types, ii) all of them encompass all methods of parsing and transmitting, iii) only the market middleware supports HFIR and asynchronous message exchange.

V. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed and compared different middleware for healthcare data exchange. This study showed the extensibility of HL7 standards and the ongoing HL7-based projects and research. The comparison analysis provided an insight into the strengths and weaknesses of the middleware architecture, the compatibility with HL7 versions, operating systems, architecture, features and connection type. Based on the survey, we found that the Mirth Connect and Mitre hData middlewares are the most appropriate for exchanging health data. This is due to the fact that they present features which are more compatible with the HL7 standard, as both middlewares are specifically meant to be used in health applications. The Websphere middleware, for general purposes, may also be used in health data exchange. However, the lack of alignment with the HL7 standard would increase the complexity of the developed solution.

In future works, we will use a HL7-based toolkit to implement, test and measure a middleware to integrate different healthcare solutions and consolidate patient data in a big-data repository using HL7 messages. Then, we will compare the results with the existing middleware and, depending on the results, propose a new HL7-based middleware architecture.

VI. REFERENCES

[1] P. Silva-Ferreira, J. Patriarca-Almeida, P. Vieira-Marques, R. Cruz-Correia, “Improving expressiveness of agents using openEHR to retrieve multi-institutional health data: Feeding local repositories through HL7 based providers.” Inf. Syst. Technol. (CISTI), 2012 7th Iber. Conf. on. IEEE, 2012, pp. 1-5.

[2] R. Cruz-Correia, J. C. Nascimento, R. D. Sousa, and H. O’Neill, “eHealth key issues in Portuguese public hospitals” Proc. - IEEE Symp. Comput. Med. Syst, 2012, pp. 1-6.

[3] L. Ribeiro, J. P. Cunha, and R. Cruz-Correia, “Information systems heterogeneity and interoperability inside hospitals: A survey,” Heal. 2010, pp. 337–343.

[4] T. Beale and S. Heard, “openEHR - Architecture Overview,” OpenEHR Found., 2007, pp. 1–79.

[5] ISO, “ISO 13606-1:2008 Health informatics — Electronic health record communication — Part 1: Reference model,” 2008. [Online]. Available: ISO 13606-1:2008(en). [Accessed: 06-Jun-2015].

[6] ISO, “ISO 13606-2:2008 Health informatics Electronic healthcare record communication Part 2: Archetype interchange specification. International Organization for Standardization,” 2008. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50119. [Accessed: 06-Jun-2015].

[7] S. Frade, S. M. Freire, E. Sundvall, J. H. Patriarca-Almeida, and R. Cruz-Correia, “Survey of openEHR storage implementations,” Proc. CBMS 2013 - 26th IEEE Int. Symp. Comput. Med. Syst., 2013, pp. 303–307.

[8] X. Ma, L. Qian, and X. Lin, “WSXP:A universal data exchange platform based on Web Services,” Inf. Sci. Eng. (ICISE), 2010 2nd Int. Conf., pp. 4815 - 4818, 2010.

[9] B. Li, L. Sun, and R. Tian, “Multi-source Heterogeneous Data Exchange System for Cloud Services Platform Based on SaaS,” Inf. Sci. Control Eng. (ICISCE), 2015 2nd Int. Conf., pp. 327-331, 2015.

[10] X. Liu, L. Ma, and Y. Liu, “A middleware-based implementation for data integration of remote devices,” Proc. - 13th ACIS Int. Conf. Softw. Eng. Artif. Intell. Networking, Parallel/Distributed Comput. SNPD 2012, pp. 219–224.

[11] S. Ngamsuriyaroj, C. Sirichamchaikul, S. Hanam, and T. Tatsanaboonya, “Patient information exchange via web services in HL 7 v3 for two different healthcare systems,” Proc. 2011 8th Int. Jt. Conf. Comput. Sci. Softw. Eng. JCSSE 2011, pp. 420–425.

[12] IBM, “IBM Message Broker 8.” [Online]. Available: http://www-01.ibm.com/support/knowledgecenter/SSKM8N_8.0.0/com.ibm.healthcare.doc.

 ISBN: 978-1-61208-438-1

[13] MITRE Corporation, “Project hData.” [Online]. Available: http://www.projecthdata.org. [Accessed: 21-Jun-2015].

[14] H. L. S. International, “Health Level Seven,” 2007. [Online]. Available: http://www.hl7.org/. [Accessed: 06-Jun-2015].

[15] ISO/IEC, “Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,” vol. 1, p. 69, 1994.

[16] L. Liu and Q. Huang, “An extensible HL7 middleware for heterogeneous healthcare information exchange,” 2012 5th Int. Conf. Biomed. Eng. Informatics, BMEI 2012, no. Bmei, 2012, pp. 1045–1048.

[17] H. C. Tung Tran, Hwa-Sun Kim, “A Development of HL7 Middleware for Medical Device Communication,” in Fifth International Conference on Software Engineering Research, Management and Applications, 2007, pp. 485–492.

[18] M. Hussain, M. Afzal, H. F. Ahmad, N. Khalid, and A. Ali, “Healthcare applications interoperability through implementation of HL7 web service basic profile,” ITNG 2009 - 6th Int. Conf. Inf. Technol. New Gener., 2009, pp. 308–313.

[19] D.-M. L. D.-M. Liou, E.-W. H. E.-W. Huang, T.-T. C. T.-T. Chen, and S.-H. H. S.-H. Hsiao, “Design and implementation of a Web-based HL7 validation system,” Proc. 2000 IEEE EMBS Int. Conf. Inf. Technol. Appl. Biomed. ITAB-ITIS 2000. Jt. Meet. Third IEEE EMBS Int. Conf. Inf. Technol, 2000, pp. 1–6.

[20] S. C. Lin, Y. L. Chiang, H. C. Lin, J. Hsu, and J. F. Wang, “Design and implementation of a HL7-based physiological monitoring system for mobile consumer devices,” Dig. Tech. Pap. - IEEE Int. Conf. Consum. Electron., 2009, pp. 1-2.

[21] N. Ibrahim, “Orthogonal classification of middleware technologies,” 3rd Int. Conf. Mob. Ubiquitous Comput. Syst. Serv. Technol. UBICOMM 2009, 2009, pp. 46–51.

[22] H. Pinus, “Middleware: Past and present a comparison,” pp. 1–5, 2004.

[23] Oracle, “Message-Oriented Middleware (MOM),” 2010. [Online]. Available: http://docs.oracle.com/cd/E19798-01/821-1798/aeraq/index.html. [Accessed: 02-Jun-2015].

[24] Microsoft, “Message Queuing (MSMQ),” 1999. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms711472(v=vs.85).aspx. [Accessed: 05-Jun-2015].

[25] IBM, “IBM MQ,” 2011. [Online]. Available: http://www-03.ibm.com/software/products/en/ibm-mq. [Accessed: 05-Jun-2015].

[26] M. Henning, “A new approach to object-oriented middleware,” IEEE Internet Comput., vol. 8, no. 1, pp. 66–75, 2004.

[27] L. Ko et al. “HL7 middleware framework for healthcare information system,” Heal. 2006 8th Int. Conf. e-Health Networking, Appl. Serv., 2006, pp. 152–156.

[28] University Health Network., “HAPI FHIR.” [Online]. Available: http://hl7api.sourceforge.net/.

[29] Inner Harbour Software, “HL7Spy.” [Online]. Available: http://hl7spy.ca/hl7-spy/. [Accessed: 21-Jun-2015].

[30] J. Reagan, “HL7 Analyst.” [Online]. Available: http://hl7analyst.codeplex.com/downloads/get/250522?releaseId=68524. [Accessed: 21-jun-2015]

[31] Meta Healthcare, “Mirth Connect - HL7 Middleware.” [Online]. Available: http://www.metahealthcare.com/solutions/mirth/. [Accessed: 21-Jun-2015].

 ISBN: 978-1-61208-438-1

Teaching Robotics and Mechanisms with MATLAB

Daniela Marghitu

Computer Science and Software Engineering Dept.
Auburn University
Auburn, Alabama 36830
Email: marghda@auburn.edu

Dan Marghitu

Mechanical Engineering Dept.
Auburn University
Auburn, Alabama 36830
Email: marghdb@auburn.edu

***Abstract*—Engineering mechanics involves the development of mathematical models of the physical world. Mechanisms and robots address the motion and the dynamics of kinematic links. MATLAB is a modern tool that has transformed mathematical methods because MATLAB not only provides numerical calculations but also facilitates analytical or symbolic calculations using the computer. The intent is to show using an R(rotation)-R(rotation)T(translation)R(rotation) chain the convenience of MATLAB for theory and applications in mechanisms. The distinction of the study from other projects is the use of MATLAB for symbolic and numerical applications. This project is intended primarily for use in dynamics of multi-body systems courses. The MATLAB graphical user interface enables students to achieve mechanisms programming helping this way the retention in engineering courses. The project can be used for classroom instruction, for self-study, and in a distance learning environment. It would be appropriate for use as an undergraduate level.**

Keywords–MATLAB; symbolic calculations; kinematic chain.

I. INTRODUCTION

In this paper, MATLAB is considered for a mathematical equations of a three moving link kinematic chain. The achievements in the robot starting with the development of the recursive Newton-Euler algorithm are given in [1]. Algorithms and equations can be developed using Kane's equations [2] and are of great value. The system kinematics are computed from experimental measurements in [3]. The method for the kinematics of rigid bodies connected by three degrees of freedom rotational joints uses the position measurements. The kinematics of a simulated three-link model and of an experimentally measured motion of human body during flight phase of a jump are discussed. The use of Mathematica and MATLAB for the analysis of mechanical systems is developed in [4] [5] [6]. Educational robotics research studies [7] and our experience of teaching robotics has shown robotics to be one of the best context for teaching computational thinking and problem solving. Habib presented the methodology of integrating MATLAB/Simulink into mechanical engineering curricula [8]. The benefits of integrating MATLAB are the basic concepts, the graphical visualization, and the mathematical libraries. MATLAB software packages for biomedical engineers including nonlinear dynamics and entropy-based methods were presented in [9] and are suitable for an upper-level undergraduate course.

This paper describes the systematic computation of motion for a three link chain using MATLAB. The software combines symbolical and numerical computations and can be applied to find and solve the motion for humans, animals, and robotic systems.

II. POSITION ANALYSIS

The planar R-RTR chain is considered is shown in Figure 1. The driver link is the rigid link 1 (the link AB). The following numerical data are given: $AB = 0.10$ m, and $CD = 0.18$ m. The angle of the driver link 1 with the horizontal axis is $\phi = 45°$. The constant angular speed of the driver link 1 is 100 rpm. A Cartesian reference frame xy is selected. The joint A is in

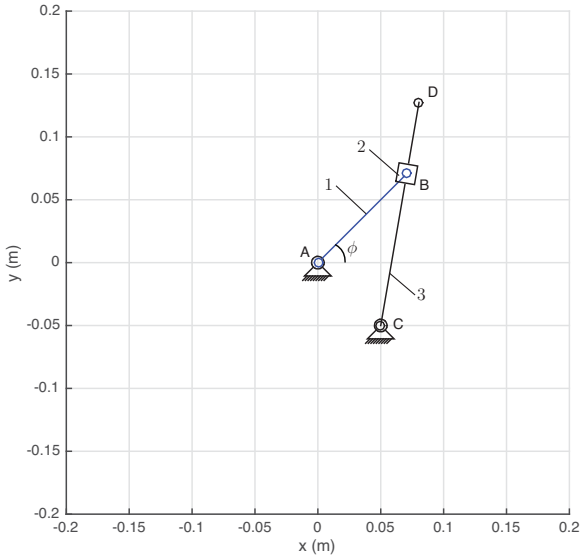


Figure 1. R-RTR chain.

the origin of the reference frame, that is, $A \equiv O$,

$$x_A = 0 \quad \text{and} \quad y_A = 0.$$

The coordinates of the joint C are given

$$x_C = 0.05 \text{ m} \quad \text{and} \quad y_C = -x_C.$$

Position of joint B

The unknowns are the coordinates of the joint B, x_B and y_B. Because the joint A is fixed and the angle ϕ is known, the coordinates of the joint B are computed from the following

 ISBN: 978-1-61208-438-1

expressions

$$x_B = AB\cos\phi,$$
$$y_B = AB\sin\phi. \quad (1)$$

The MATLAB commands for joints A, C, and B are:

```
AB = 0.10 ;  % (m)
CD = 0.18 ;  % (m)
phi = pi/4;  % (rad)
xA = 0; yA = 0;
rA_ = [xA yA 0];
xC =  0.05;  % (m)
yC = -xC;
rC_ = [xC yC 0];
xB = AB*cos(phi);
yB = AB*sin(phi);
rB_ = [xB yB 0];
```

Angle ϕ_2
The angle of link 2 (or link 3) with the horizontal axis is calculated from the slope of the straight line BC:

$$\phi_2 = \phi_3 = \arctan\frac{y_B - y_C}{x_B - x_C}. \quad (2)$$

Position of joint D
The unknowns are the coordinates of the joint D, x_D and y_D

$$x_D = x_C + CD\cos\phi_3,$$
$$y_D = y_C + CD\sin\phi_3. \quad (3)$$

The MATLAB commands for the position of D are:

```
phi2 = atan((yB-yC)/(xB-xC));
phi3 = phi2;
CB = norm([xB-xC, yB-yC]);
ux = (xB - xC)/CB;
uy = (yB - yC)/CB;
xD = xC + CD*ux; yD = yC + CD*uy;
rD_ = [xD yD 0];
```

The components of the unit vector of the vector $\overrightarrow{CD}$ are `ux` and `uy`. The numerical results are:

```
% phi = phi1 = 45 (degrees)
% rA_ = [ 0.000, 0.000,0] (m)
% rC_ = [ 0.050,-0.050,0] (m)
% phi2 = phi3 = 80.264 (degrees)
% rD_ = [ 0.080, 0.127,0] (m)
```

The position simulation for a complete rotation of the driver link 1 ($\phi \in [0, 360°]$) is obtained with the MATLAB using a loop command and the graphical representation is shown in Figure 2.

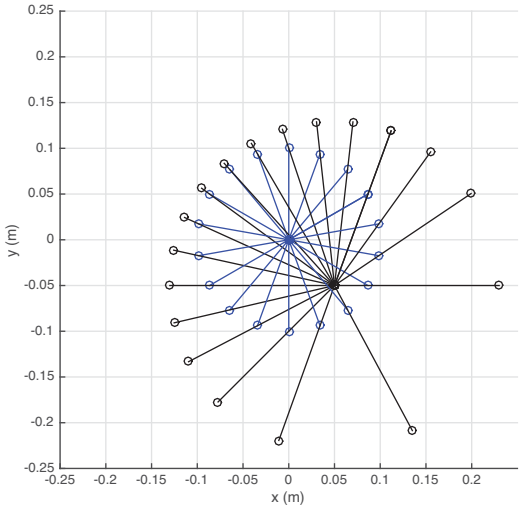


Figure 2. MATLAB representation of the R-RTR chain for a complete rotation of the driver link.

III. Velocity and Acceleration Analysis

The velocity of the point B_1 on the link 1 is

$$\mathbf{v}_{B_1} = \mathbf{v}_A + \boldsymbol{\omega}_1 \times \mathbf{r}_{AB} = \boldsymbol{\omega}_1 \times \mathbf{r}_B, \quad (4)$$

where $\mathbf{v}_A \equiv \mathbf{0}$ is the velocity of the origin $A \equiv O$.
The angular velocity of link 1 is

$$\boldsymbol{\omega}_1 = \omega_1\,\hat{\mathrm{k}} = \frac{\pi n}{30}\,\hat{\mathrm{k}} = \frac{\pi(100)}{30}\,\hat{\mathrm{k}}\ \ \mathrm{rad/s}. \quad (5)$$

The position vector of point B is

$$\mathbf{r}_{AB} = \mathbf{r}_B - \mathbf{r}_A = x_B\,\hat{\mathrm{i}} + y_B\,\hat{\mathrm{j}} + z_B\,\hat{\mathrm{k}}. \quad (6)$$

The velocity of point B_2 on the link 2 is $\mathbf{v}_{B_2} = \mathbf{v}_{B_1}$ because between the links 1 and 2 there is a rotational joint. The velocity of $B_1 = B_2$ is

$$\mathbf{v}_{B_1} = \mathbf{v}_{B_2} = \begin{vmatrix} \hat{\mathrm{i}} & \hat{\mathrm{j}} & \hat{\mathrm{k}} \\ 0 & 0 & \omega \\ x_B & y_B & 0 \end{vmatrix}. \quad (7)$$

The acceleration of the point $B_1 = B_2$ is

$$\begin{aligned} \mathbf{a}_B = \mathbf{a}_{B_1} &= \mathbf{a}_{B_2} = \mathbf{a}_A + \boldsymbol{\alpha}_1 \times \mathbf{r}_B + \boldsymbol{\omega}_1 \times (\boldsymbol{\omega}_1 \times \mathbf{r}_B) \\ &= -\boldsymbol{\omega}_1^2 \mathbf{r}_B. \end{aligned} \quad (8)$$

The angular acceleration of link 1 is $\boldsymbol{\alpha}_1 = \dot{\boldsymbol{\omega}}_1 = \mathbf{0}$. The MATLAB commands for the velocity and acceleration of $B_1 = B_2$ are

```
vB1_ = vA_ + cross(omega1_,rB_);
vB2_ = vB1_;
aB1_ = aA_  + cross(alpha1_,rB_) - ...
        dot(omega1_,omega1_)*rB_;
aB2_ = aB1_;
```

The numerical results are:

```
% vB1_ = vB2_ =[-0.740, 0.740,0] (m/s)
% aB1_ = aB2_ =[-7.754,-7.754,0] (m/s^2)
```

The velocity of the point B_3 on the link 3 is calculated in terms of the velocity of the point B_2 on the link 2

$$\mathbf{v}_{B_3} = \mathbf{v}_{B_2} + \mathbf{v}_{B_{32}}^{rel} = \mathbf{v}_{B_2} + \mathbf{v}_{B_{32}}, \quad (9)$$

where $\mathbf{v}_{B_{32}}^{rel} = \mathbf{v}_{B_{32}}$ is the relative acceleration of B_3 with respect to a reference frame attached to link 2. This relative velocity is parallel to the sliding direction BC, $\mathbf{v}_{B_{32}} || BC$, or

$$\mathbf{v}_{B_{32}} = v_{B_{32}}\cos\phi_2\hat{\mathrm{i}} + v_{B_{32}}\sin\phi_2\hat{\mathrm{j}}, \quad (10)$$

where ϕ_2 is known from position analysis. The points B_3 and C are on the link 3 and

$$\mathbf{v}_{B_3} = \mathbf{v}_C + \boldsymbol{\omega}_3 \times \mathbf{r}_{CB} = \boldsymbol{\omega}_3 \times (\mathbf{r}_B - \mathbf{r}_C), \quad (11)$$

 ISBN: 978-1-61208-438-1

where $\mathbf{v}_C \equiv \mathbf{0}$ and the angular velocity of link 3 is $\boldsymbol{\omega}_3 = \omega_3\hat{\mathrm{k}}$. Equations (9), (10), and (11) give

$$\begin{vmatrix} \hat{\mathrm{i}} & \hat{\mathrm{j}} & \hat{\mathrm{k}} \\ 0 & 0 & \omega_3 \\ x_B - x_C & y_B - y_C & 0 \end{vmatrix} = \mathbf{v}_{B_2} + v_{B_{32}} \cos\phi_2 \hat{\mathrm{i}} + v_{B_{32}} \sin\phi_2 \hat{\mathrm{j}}. \quad (12)$$

Equation (12) represents a vectorial equations with two scalar components on x-axis and y-axis and with two unknowns ω_3 and $v_{B_{32}}$

$$\begin{aligned} -\omega_3(y_B - y_C) &= v_{Bx} + v_{B_{32}} \cos\phi_2, \\ \omega_3(x_B - x_C) &= v_{By} + v_{B_{32}} \sin\phi_2. \end{aligned} \quad (13)$$

The vectorial equation (12) is obtained in MATLAB with:

```
omega3z=sym('omega3z','real');
vB32=sym('vB32','real');
omega3u_ = [ 0 0 omega3z];
vB3B2u_ = vB32*[cos(phi2) sin(phi2) 0];
vC_ = [0 0 0]; % C is fixed
vB3_ = vC_ + cross(omega3u_,rB_-rC_);
eqvB_ = vB3_ - vB2_ - vB3B2u_;
eqvBx = eqvB_(1); eqvBy = eqvB_(2);
```

The solutions of the system are obtained with:

```
solvB = solve(eqvBx,eqvBy);
omega3zs=eval(solvB.omega3z);
vB32s=eval(solvB.vB32);
omega3_ = [0 0 omega3zs];
omega2_ = omega3_;
vB3B2_ = vB32s*[cos(phi2) sin(phi2) 0];
```

The numerical results are:

```
% omega2_ = omega3_ = [0,0, 6.981](rad/s)
% vB32_ = [-0.102,-0.596,-0] (m/s)
```

The acceleration of the point B_3 on the link 3 is calculated in terms of the acceleration of the point B_2 on the link 2

$$\mathbf{a}_{B_3} = \mathbf{a}_{B_2} + \mathbf{a}^{rel}_{B_3B_2} + \mathbf{a}^{cor}_{B_3B_2} = \mathbf{a}_{B_2} + \mathbf{a}_{B_{32}} + \mathbf{a}^{cor}_{B_{32}}, \quad (14)$$

where $\mathbf{a}^{rel}_{B_3B_2} = \mathbf{a}_{B_{32}}$ is the relative acceleration of B_3 with respect to B_2 on link 3. This relative acceleration is parallel to the sliding direction BC, $\mathbf{a}_{B_{32}} || BC$, or

$$\mathbf{a}_{B_{32}} = a_{B_{32}} \cos\phi_2 \hat{\mathrm{i}} + a_{B_{32}} \sin\phi_2 \hat{\mathrm{j}}. \quad (15)$$

The Coriolis acceleration of B_3 realative to B_2 is

$$\begin{aligned} \mathbf{a}^{cor}_{B_{32}} &= 2\ \boldsymbol{\omega}_3 \times \mathbf{v}_{B_{32}} = 2\ \boldsymbol{\omega}_2 \times \mathbf{v}_{B_{32}} \\ &= 2 \begin{vmatrix} \hat{\mathrm{i}} & \hat{\mathrm{j}} & \hat{\mathrm{k}} \\ 0 & 0 & \omega_3 \\ v_{B_{32}} \cos\phi_2 & v_{B_{32}} \sin\phi_2 & 0 \end{vmatrix} = \\ &2(-\omega_3 v_{B_{32}} \sin\phi_2 \hat{\mathrm{i}} + \omega_3 v_{B_{32}} \cos\phi_2 \hat{\mathrm{j}}). \end{aligned} \quad (16)$$

The points B_3 and C are on the link 3 and

$$\mathbf{a}_{B_3} = \mathbf{a}_C + \boldsymbol{\alpha}_3 \times \mathbf{r}_{CB} - \omega_3^2 \mathbf{r}_{CB}, \quad (17)$$

where $\mathbf{a}_C \equiv \mathbf{0}$ and the angular acceleration of link 3 is $\boldsymbol{\alpha}_3 = \alpha_3\hat{\mathrm{k}}$. Equations (14), (15), (16), and (17) give

$$\begin{aligned} &\begin{vmatrix} \hat{\mathrm{i}} & \hat{\mathrm{j}} & \hat{\mathrm{k}} \\ 0 & 0 & \alpha_3 \\ x_B - x_C & y_B - y_C & 0 \end{vmatrix} - \omega_3^2(\mathbf{r}_B - \mathbf{r}_C) \\ &= \mathbf{a}_{B_2} + a_{B_{32}}(\cos\phi_2 \hat{\mathrm{i}} + \sin\phi_2 \hat{\mathrm{j}}) + 2\ \boldsymbol{\omega}_3 \times \mathbf{v}_{B_{32}}. \end{aligned} \quad (18)$$

Equation (18) represents a vectorial equations with two scalar components on x-axis and y-axis and with two unknowns α_3 and $a_{B_{32}}$

$$\begin{aligned} &-\alpha_3(y_B - y_C) - \omega_3^2(x_B - x_C) \\ &= a_{Bx} + a_{B_{32}} \cos\phi_2 - 2\omega_3 v_{B_{32}} \sin\phi_2, \\ &\alpha_3(x_B - x_C) - \omega_3^2(y_B - y_C) \\ &= a_{By} + a_{B_{32}} \sin\phi_2 + 2\omega_3 v_{B_{32}} \cos\phi_2. \end{aligned} \quad (19)$$

The MATLAB commands for the calculating α_3 and $a_{B_{32}}$ are:

```
aB3B2cor_ = 2*cross(omega3_,vB3B2_);
alpha3z=sym('alpha3z','real');
aB32=sym('aB32','real');
alpha3u_ = [0 0 alpha3z];
aB3B2u_ = aB32*[cos(phi2) sin(phi2) 0];
aC_ = [0 0 0]; % C is fixed
aB3_=aC_+cross(alpha3u_,rB_-rC_)- ...
dot(omega3_,omega3_)*(rB_-rC_);
eqaB_ = aB3_ - aB2_ - aB3B2u_ - ...
aB3B2cor_;
eqaBx = eqaB_(1); eqaBy = eqaB_(2);
solaB = solve(eqaBx,eqaBy);
alpha3zs=eval(solaB.alpha3z);
aB32s=eval(solaB.aB32);
alpha3_ = [0 0 alpha3zs];
alpha2_ = alpha3_;
aB32_ = aB32s*[cos(phi2) sin(phi2) 0];
```

and the vectorial solutions are:

```
% alpha2_=alpha3_=[0,0,-17.232](rad/s^2)
% aB3B2_=[ 0.505, 2.942,0] (m/s^2)
```

The velocity of D is

$$\mathbf{v}_D = \mathbf{v}_C + \boldsymbol{\omega}_3 \times \mathbf{r}_{CD} = \boldsymbol{\omega}_3 \times (\mathbf{r}_D - \mathbf{r}_C) = \begin{vmatrix} \hat{\mathrm{i}} & \hat{\mathrm{j}} & \hat{\mathrm{k}} \\ 0 & 0 & \omega_3 \\ x_D - x_C & y_D - y_C & 0 \end{vmatrix}. \quad (20)$$

The acceleration of D is

$$\begin{aligned} &\mathbf{a}_D = \mathbf{a}_C + \boldsymbol{\alpha}_3 \times \mathbf{r}_{CD} - \omega_3^2 \mathbf{r}_{CD} = \\ &\boldsymbol{\alpha}_3 \times (\mathbf{r}_D - \mathbf{r}_C) - \omega_3^2(\mathbf{r}_D - \mathbf{r}_C) = \\ &\begin{vmatrix} \hat{\mathrm{i}} & \hat{\mathrm{j}} & \hat{\mathrm{k}} \\ 0 & 0 & \alpha_3 \\ x_D - x_C & y_D - y_C & 0 \end{vmatrix} \\ &-\omega_3^2 \left[(x_D - x_C)\hat{\mathrm{i}} + (y_D - y_C)\hat{\mathrm{j}}\right]. \end{aligned} \quad (21)$$

The velocity and acceleration of D are calculated in MATLAB with:

```
vD_ = vC_ + cross(omega3_,rD_-rC_);
aD_=aC_+cross(alpha3_,rD_-rC_)-...
dot(omega3_,omega3_)*(rD_-rC_);
```

IV. Dynamic Force Analysis

The force of gravity, $\mathbf{G}_1$, the force of inertia, $\mathbf{F}_{in1}$, and the moment of inertia, $\mathbf{M}_{in1}$, of the link 1 are calculated using the MATLAB commands:

```
m1 = rho*AB*h*d;
G1_ = [0,-m1*g,0];
Fin1_ = -m1*aC1_;
IC1 = m1*(AB^2+h^2)/12;
IA  = m1*(AB^2+h^2)/3 ;
alpha1_ = [0 0 0];
Min1_ = -IC1*alpha1_;
```

For the links 2 and 3 there are similar MATLAB statements. The joint forces are calculated using Newton-Euler equations of motion for each link, as depicted in Figure 3. The dynamic

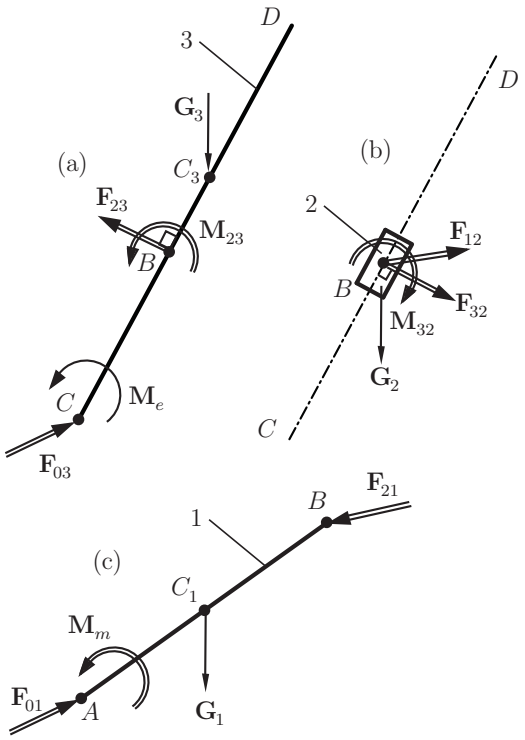


Figure 3. Free body diagrams for each link.

force analysis starts with the last link 3 because the given external moment acts on this link. For link 3, the unknowns are:

```
F23x=sym('F23x','real');
F23y=sym('F23y','real');
F23_=[F23x, F23y, 0];
M23z=sym('M23z','real');
M23_=[0, 0, M23z];
```

Because the joint between 2 and 3 at B is a translational joint the reaction force `F23_` is perpendicular to the sliding direction BC: `eqF23 = (rB_-rC_)*F23_.';` A moment equation for all the forces and moments that act on link 3 with respect to the fixed point C can be written:

```
eqM3C_ = cross(rB_-rC_,F23_)+....
cross(rC3_-rC_,G3_)+M23_+Me_-IC*alpha3_;
eqM3z = eqM3C_(3);
```

There are two scalar equations `%(1)` and `%(2)` with three unknowns `F23x, F23y, M23z`. The force calculation will continue with link 2. For link 2, a new unknown force, the reaction of link 1 on link 2 at B, is introduced:

```
F12x=sym('F12x','real');
F12y=sym('F12y','real');
F12_=[F12x, F12y, 0];
```

The Newton-Euler equations of motion for link 2, are written as:

```
eqF2_ = F12_+G2_+(-F23_)-m2*aC2_;
eqF2x = eqF2_(1);
eqF2y = eqF2_(2);
eqM2_ = -M23_ - IC2*alpha2_;
eqM2z = eqM2_(3);
```

Now there are 5 equations with 5 unknowns and the system can be solve using MATLAB:

```
sol23=solve(eqF23,eqM3z,eqF2x,eqF2y,eqM2z);
F23xs=eval(sol23.F23x);
F23ys=eval(sol23.F23y);
F12xs=eval(sol23.F12x);
F12ys=eval(sol23.F12y);
M23zs=eval(sol23.M23z);
```

The motor moment, `M_`, required for the dynamic equilibrium of the mechanism is determined from:

```
Mm_=IA*alpha1_-(cross(rC1_,G1_)+...
cross(rB_,-F12s_));
```

V. Conclusion

The paper presented a modern tool to teach robotic systems using MATLAB. MATLAB provides numerical and symbolical calculations for an R-RTR kinematic chain. This study is intended for use in mechanism and robotic courses for the undergraduate level.

References

[1] R. Featherstone and D. Orin, "Robot dynamics: Equations and algorithms," in IEEE International Conference Robotics & Automation, San Francisco, CA, 2000, pp. 826–834.

[2] T. Kane and D. Levinson, "Kane's dynamical equations in robotics," International Journal of Robotics Research, vol. 2(3), 1983, pp. 3–21.

[3] J. Lee, H. Flashner, and J. McNitt-Gray, "Estimation of multibody kinematics using position measurements," Journal of Computational and Nonlinear Dynamics, vol. 6(3), 2011, p. 9 pages.

[4] D. Marghitu, Mechanical Engineer's Handbook. Academic Press, San Diego, CA, 2001.

[5] ——, Kinematic Chains and Machine Component Design. Elsevier, Amsterdam, 2005.

[6] D. Marghitu and M. Dupac, Advanced Dynamics: Analytical and Numerical Calculations with MATLAB. Springer, 2009.

[7] J. Davis, B. Wellman, M. Anderson, and R. M., "Providing robotic experiences through object-based programming (preop)," in Proceedings of 2009 Alice Symposium in cooperation with SIGCSE, ACM, Durham, NC, 2009.

[8] M. Habib, "Enhancing mechanical engineering deep learning approach by integrating matlab/simulink," Int. J. Engng. Ed., vol. 21, 2005, pp. 906–914.

[9] J. Semmlow and B. Griffel, Biosignal and Medical Image Processing. CRC Press, 2014.

 ISBN: 978-1-61208-438-1

Case of Enterprise Architecture in Manufacturing Firm

Alicia Valdez, Griselda Cortes, Sergio Castaneda
Research Center
Autonomous University of Coahuila
Coahuila, Mexico
Email: aliciavaldez@uadec.edu.mx,
griselda.cortes@uadec.edu.mx,
sergiocastaneda@uadec.edu.mx

Gerardo Haces, Jose Medina
Research Center
Autonomous University of Tamaulipas
Tamaulipas, Mexico
Email: ghaces@uat.edu.mx, jmedina@uat.edu.mx

***Abstract*— The small and medium enterprises (SME's) have a low level of survival and are facing serious problems like access to financing, weak management capacity, poor information about market opportunities, limited information about access to innovation and research funds, new technologies and methods of work organization, the integration of its key processes into information and communication technologies (ICT's). The methodology of Enterprise Architecture (EA) can provide a model of integration supported with strategic planning, integrated by partial architectures like business, data, applications and technology. In this study case, we are providing information that was collected from different sources of a medium manufacturing firm to design and implement an EA. As a result we get to the identification of strategic changes supported by the methodology, the assessment of option for change, and a change plan for the adoption of the methodology in their processes; those processes must improve their competitiveness and productivity.**

Keywords-Enterprise architecture; Strategic planning; Manufacturing companies.

I. INTRODUCTION

The EA is a methodology which is looking to provide a framework for the companies, for the use of the information in the processes of the business [1]. EA consist of methodological frameworks, architectural frameworks, technologies and standards [2]. The identification of the key processes of the companies in the manufacturing sector located in the areas of marketing, engineering, logistics, productivity and business management, will provide important guidelines for the design of the EA. Studies conducted by researchers in Germany indicate that the administration of the EA is a factor that leads to changes in the companies. They found the impact of 5 dimensions to take advantage of the benefits derived from the EA [3]:

- Quality of the EA
- Quality of the infrastructure of the EA
- Quality of service of EA
- Culture of EA
- Use of the EA

The five dimensions are important to achieve a successful EA, with the realization of benefits measured by the EA Benefit Realization Model (EABRM) [3].

EA is used as a strategic tool and a mechanism to support the alignment between business strategy and ICT's. It also represents a change in the organizational learning, and not just a change of intellectual learning, but learning to cooperate and share information across the entire enterprise [4]. It enables companies to achieve a logical balance between technological efficiency and innovation seeking competitive advantages. Therefore it is necessary to create strategic solutions to enhance the capabilities of enterprises and respond with agility to the challenges, be they business or technology, that today's markets require.

The SME's are major generator of employment in Mexico (7 of 10 jobs are provided by them) [5]. The design and implementation of EA in support of the strategy of SME's in the manufacturing business sector, will increase the competitiveness and productivity of these companies.

The main purpose of this article is designing and implementing an EA for SME's for the manufacturing sector that support the innovation and competitiveness in the national and global market through methodologies, frameworks, technologies, and, standards. This research can be replicated in SME's in the manufacturing sector of metalworking.

This paper is structured as follows: In Section II the state of the art is presented. In Section III, partial architectures, business, application, and technology are described. Also, the values of the company for the study are stated. Section IV describes the principal findings of the study application of each partial architecture and their final recommendations. Section V presents conclusion and future work.

 ISBN: 978-1-61208-438-1

II. STATE OF THE ART

Porter defines innovation as the creation of new products, processes, knowledge or services by using existing, or new scientific, or technical knowledge. Furthermore, Porter also mentions that firms are evolving into value chains based on intangible assets, such as knowledge, technology, intellectual property, and others [6].

Afua also defines "innovation as the use of new technological knowledge, market knowledge, and business models that can deliver a new product, or service, or product/service combinations to customers who will purchase at prices that will provide profits" [7].

Consequently, the "Innovation Economy" involves the production and generation of knowledge, and its application in products, processes, and services. Thus, it has become the main asset for developing dynamic competitive advantages [8].

Innovation is the most important topic required for companies to grow successfully. In addition, some factors are changing the environment where the companies are competing. Among these factors include [9]:

- Access to Knowledge: Permit to companies has access to the best ideas, technologies, research resources, and experts at low cost.
- Trade Barriers: Are rapidly being dismantled, thus opening up all markets to global suppliers.
- Access to Capital: The funds may now seek opportunities on a global basis, and companies must compete internationally for capital.
- Technological Obsolescence: Market life cycles are now less than product development cycles. In addition, companies are developing new methods to reduce their product development times.

A study by the Centro de Tecnologia Avanzada A.C. (CIATEQ) developed in 2008, found insufficient capacity of SME's for the production of knowledge and technology, little demand for them, and the disconnection between demand and possible public offering [10]. The study reflects that SME's had greater difficulties in order to integrate into the productive chains. In addition, there is a lack of coordination between the system of technological innovation and the technological demand from firms.

In the adoption of new methods for the organization of work and innovation, the EA methodology was placed into consideration. EA is a methodology that aims to provide companies with a framework for the use of the information in business processes in ways that support their business strategy [4], and provides the strategic alignment between the business strategy and the ICT's.

Some frameworks have been created for providing a guide or method for the establishment of the EA. These are:

- The Zachman Framework.
- The Department of Defense Architecture Framework (DoDAF).
- The Open Group Architecture Framework (TOGAF).

The Zachman Framework [11] was created by John Zachman in the early 60's at the International Business Machines (IBM) Corporation. Consequently, he developed the framework to define the information systems. This combines rows and columns that represent the perspectives, views, and descriptions types. The perspectives include scope, business model, systems model, technology model, detailed representations, and company's performance.

The description types are data, function, network, people, time, and motivation. Therefore, Figure 1 shows the Zachman Framework with its elements. Each cell contains a set of elements that represent diagrams or documents on the specific architecture and the level of details. For example, in the column of the functions or processes with line of objectives and scope, the cell has a list of processes which run the business.

Ylimaki and Halttunen say that all the columns and rows are important because they form the abstractions of the company [12]. Thus, each cell must include a primitive graphical model which describes the company from the viewpoint of the perspective that it is analyzing.

The Zachman Framework established the basis for the next generations of frameworks. This includes DoDAF, which later became TOGAF, which is based on the Architectural Development Method (ADM).

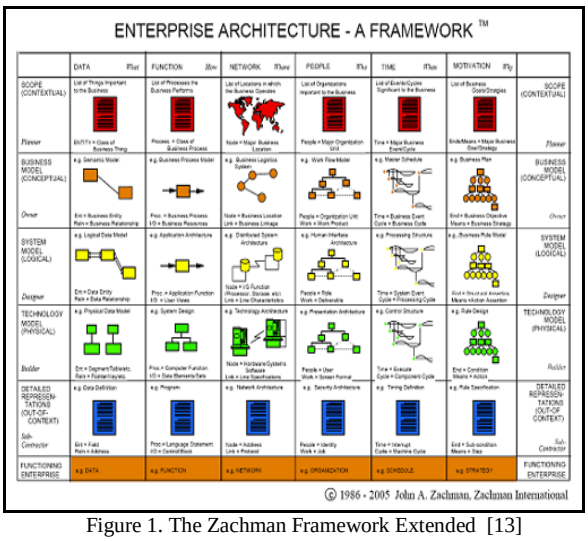


Figure 1. The Zachman Framework Extended [13]

In Figure 2, nine phases were represented. Therefore, these phases are preliminary analysis, architecture vision, business architecture, information systems architectures, technology architecture, opportunities and solutions, migration plan, implementation of governance, and architecture change management. All these components of TOGAF generate deliverables in the form of diagrams, flowcharts, structures, definitions, and other artifacts.

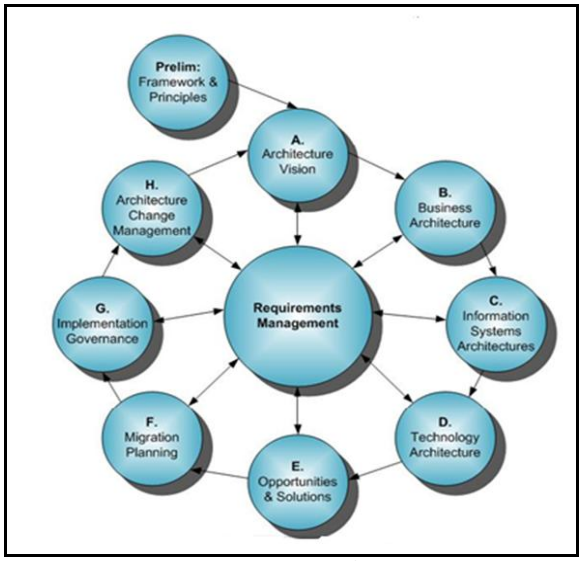


Figure 2. Phases of TOGAF [14]

Every framework has a common factor. They aim to empower the company through the ICT's search which helps to increase the productivity and the competitiveness of the company.

With the increment of complexity in the companies, the needs of data processing are increasing [13]. In this case, references about TOGAF and Zachman Framework has been taken [11]. This is used to construct the design proposal of the architecture. In addition, software designing tools like Essential Architecture Manager and the editor Protégé Ontology Editor Version 3.4, have been used in the management of the data in the architecture design.

Other researchers have developed advanced applications of EA, as Bernard [15] who has defined EA as a holistic management, planning, and documentation activity, and has introduced the EA Cube Framework and implementation methodology. Where lines of business were defined as five sub-architectures: Strategic initiatives, business services, information flows, systems and applications, and technology infrastructure.

"Newer approaches as business services, exemplifies how EA can link strategy, business, and technology components across the enterprise within a service bus that encompasses platform independent horizontal and vertical EA components" [15].

Ahlemann, Stettiner, Messerschmidt, and Legner establish that the Enterprise Architecture Management (EAM) is a driver of strategic architecture initiatives as: The implementation of reference models and industry norms, with the goal of adopting best practices, standardization and harmonization, with the goal of reducing the heterogeneity and complexity of business processes, applications, data, infrastructure technology, service oriented and modularization , with the goal of creating reusable services and modules [16].

III. PARTIAL ARCHITECTURES

Research and studies in Europe indicate that the EA is a driver for transformations in companies [3]. Orantes, Gutierrez and Lopez mentioned that companies should be constantly evolving, redefining business processes to achieve business architecture (BA) which is the basis for subsequent architectures [2]. The data flow of the study is shown in Figure 3.

With these premises, partial architectures have been constructed. However, the first is the BA.

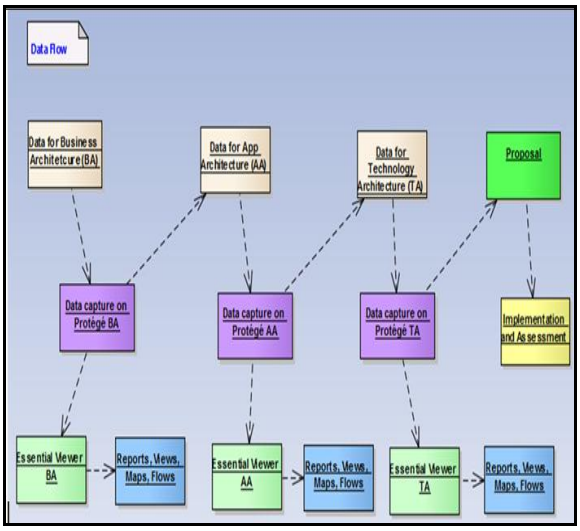


Figure 3. Data flow of the EA

A. Business Architecture

The purpose of the BA is to define the business, document organizational structure, identify and define business functions and processes, relying on strategic planning with their areas of interest [17]. The BA involves some elements of the company like the mission, vision, objectives, goals, values and policies; business processes, procedures and functions, and organizational structure, situational analysis, customers, markets, products, and long, medium, and short strategies.

 ISBN: 978-1-61208-438-1

The company of the study case is a medium enterprise that provides raw material to the large steel companies in northeast of Mexico. This company was established in 1982 to meet the needs of the industry in the manufacture and machining of metal parts. Consequently, the machined parts are made through computerized numerical control machines (CNC). The main products which were manufactured are generally forklift parts, rotating joints, plates thousand holes (clamps, screws, etc.), and various mechanical equipment parts and assembly work. The company has 65 employees. Figure 4 presents some data for the BA.

The structure of the company has 4 levels, corresponding to the CEO and Sales manager at the top position. As follows, these levels include Head of production machining, Head buyer, Head finance, and Human resource manager. Other levels have Supervisors for machining and pailer areas.

Figure 4. Elements of the Business Architecture

The company competes in the regional market of Mexico, and has a local quality certification. Their strategy is to produce high quality metals that the markets are demanding for.

B. Applications Architecture

The Applications Architecture (AA) contains the software products that the company has for the support of the processes. The objectives, principles, and capabilities that govern this architecture are presented in Table I.

The objective of AA is defined as the best kinds of applications to manage data and support business processes with the minimum packaged applications. As a result, the capabilities for the management of the AA are the analysis, design, programming and implementation of information systems, search packaged solutions tailored towards the needs of the SME's, and providing technical support for software and hardware in all the company. Some of the current applications are presented in Table II. In this case, the applications are related to the processes that supports the firm.

TABLE I. BASIS FOR THE APPLICATIONS ARCHITECTURE

AA	Name	Description
Objective	Define the best kinds of applications to manage data and support business processes.	Define the best applications that support the business processes.
Principle	Customizing minimum packaged applications.	Minimize app package, customization will improve the ability to ensure ongoing maintenance and maximum value obtained from the adoption of a package solution.
Capabilities	Analysis, design, programming and implementation of information systems. Search packaged solutions tailored to the needs of the SME's Provide technical support for software and hardware throughout the company.	Domain in the analysis, design, programming and implementation of information systems. Domain in search packaged solutions tailored to the needs of the SME's Domain to provide technical support for software and hardware.

TABLE II. VALUES FOR INSTANCES OF APPLICATIONS ARCHITECTURE

Name	Description	Domain of App	Performed by business processes
Stock Information System	Management of the inputs and outputs of the company general store.	Update catalog of items, articles inventory processing.	Registry inputs and outputs of goods and raw materials.
Quality	Spreadsheets records quality of finished products.	Data of finished products according to production plan.	Verify the manufacturing process according specifications with production.
Client IS	Manage Client Portfolio.	Update Clients Portfolio, Electronic Billing.	Client Portfolio.
Financial IS	General Financial System.	Update chart of accounts, sub and sub-sub. Update Cost-Centers.policies for debit and credit accounts. Update the information of credit banks.	Accounting Manager.

C. Technology Architecture

The Technology Architecture (TA) represents the computational equipment that supports the applications for the operation of the company. The objectives,

 ISBN: 978-1-61208-438-1

principles, and capabilities of TA are displayed in Table III.

The consolidation of the technology infrastructure is the main objective, with minimun diversity for maintenance purposes. The capabilities are complex because TA manages communications and networks. Also, it provides technical support and various services like platforms integration and monitoring. Thus, Figure 5 presents the equipment.

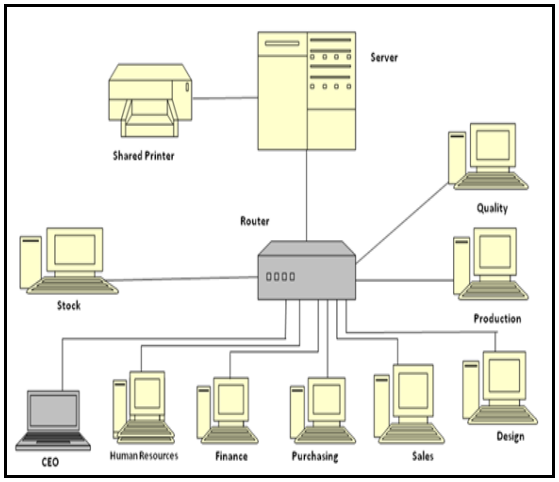


Figure 5. The computational equipment

The company has one server with 9 computers in the local area network. However, one shared printer provides the printing services. Other areas as quality, purchasing, sales, production, design, finances, human resources, and the CEO are supported by the equipment.

TABLE III. BASIS FOR THE TECHNOLOGY ARCHITECTURE

TA	Name	Description
Technology Principle	Minimun diversity of technological products	
Objective	Technology infrastructure consolidation	The technology infrastructure will be consolidated in the company
Principle	Minimum diversity of technology products	Minimum diversity for better maintenance of equipment
Capabilities	Software management services, hardware platform services, security services, technical support, communication and networking services	Ability to acquire, install and configure networking and communication. Ability to detect and correct faults in computer equipment. Ability to manage all software. Ability to manage hardware platforms.

Therefore, one equipment can be used to run various applications.

Table IV summarizes the results of the architectures integration with the processes, applications, and technology that supports the application.

TABLE IV. RESULTS OF THE INTEGRATION

Areas	Process 1	Process 2	Process 3	Process 4	Application	Technology
Quality	Controls of imput materials, components and cosumables	Verify the production plan meets especifications	Testing and inspection using ultrasonic methods or industrial inspection	Identify causes of non.confomity in items or lots and take corrective action	**Spreadsheets**	**Computer connected to LAN and internet access**
Training	Planning and monitoring The Training Business Plan	Detect training needs of business areas, especially productive areas			**There is no application for this process**	**There is no technology for this process**
Shipments	Shipment management of finished products				**There is no application for this process**	**There is no technology for this process**
Billing	Billing management				**Billing information system**	**Computer connected to LAN and internet access**
Production/ Machining and Pailer	Program production cycles	Cutting, marking, machining and forming plates and steel profiles	Management of fabrication process	Calibrate equipment periodically	**Production information system**	**Computer connected to LAN and internet access**
Sales	Sales management	Detecting customer needs			**Clients Information System**	**Computer connected to LAN and internet access**

IV. RESULTS

After the design of the partial architectures in this firm, the results show that two processes are not completely supported by applications and technology. These include Training and Shipment.

E-commerce which could improve the sales volume and increase productivity is highly recommended when rethinking a strategic planning of the company. Therefore, quality and production processes are supported by spreadsheets.

The Microsip Manufacturing System [18] is a solution designed for SME's that do not have applications in the production areas. Also, the study of EA confirms the need that had already been detected to automate production processes. Thus, the advantages of the software are:

- Planning and control of production.
- Calculation of production.
- Prioritize work orders.
- Monitoring of workers.
- Monitoring of machines.
- Simulation of production orders.
- Production reports.

 ISBN: 978-1-61208-438-1

- Tracking production.
- Dynamic queries with access to all production data.
- Control of shipments.
- Conversion to graphics.
- Integration with other installed modules of Microsip brand.

With this special solution, SME's control of the shipments of finished products will be achieved.

Information systems are used in training areas in order to manage the main elements, such as courses, facilitators, identifying the needs of the plant for training, and the proposed dates.

The activities of internal or external training generate a dynamic company that leads to improved productivity and competitiveness.

Regarding the proposal to integrate e-commerce firm on a long term, the following activities were proposed:

- Development of a dynamic website to promote the manufactured products.
- Investment in ICT's could increase competitiveness and growth into new domestic and international markets.
- Establish a definition of roles, functions, and policies for recategorizing.
- Get an open line of credit to finance the expansion plan and the purchase of equipment and machinery.
- Competition is intense in the sector of this industry. Thus, achieving a better position in the market is necessary to identify and select new potential customers, increase advertising in all media, expand their sales channels, and follow up through customer service.
- Fostering a culture of total quality in the whole company.
- The change plan is long term, and it includes management activities to close gaps encountered and the acquisition and implementation of information systems and technology.

In addition, this project helps in meeting the needs of SME's companies to propose affordable solutions that use business management resources and technology, to solve problems. This study is limited to manufacturing SME's.

Other findings in terms of improvement show that SME's have demonstrated alignment with business strategy to drive a strong organizational culture and technological infrastructure.

V. CONCLUSION AND FUTURE WORK

In this paper, the EA methodology was designed and applied in a medium manufacturing company with information provided from different sources of the firm, where the applications and technology that support their business processes were analyzed, linking them through tables and charts. As a result we found that the company requires an ICT's investment in production and quality processes, which would integrate the information that is obtained in real time and would expedite the processes of decision-making, as well as the integration of e-commerce to the sales strategy. This can help increase the productivity of the company.

The advantage of this approach lies in that the company is analyzed holistically, especially in its core processes.

Given the importance for EA, organizations will increasingly support their EA efforts as a virtual reporting structure with collaboration of all people to deliver substantial business value.

REFERENCES

[1] S. Spewak and S. Hill, "Enterprise architecture planning, developing a blueprint for data, application and technology", Wiley publisher, USA, 1992, pp. 1-6.

[2] S. Orantes, A. Gutierrez, and M. Lopez," Enterprise architectures: Business processes management vs Services oriented architectures, are they related?", Redalyc, 2009, pp. 136-144, vol. 13.

[3] M. Lange, J. Mandling, and J. Recker, "Realizing benefits from enterprise architecture: A measurement model" , ECIS 2012 Proceedings, 2012, A.E. Library, Barcelona, Spain.

[4] J. Poutanen, "The social dimension of enterprise architecture in government", Journal of Enterprise Architecture, 2012, pp. 19-29, vol 8.

[5] Secretary of Economy, "SMEs news",http://economia.gob.mx/ [retrieved: 04-2013].

[6] M. Porter, "The Competitive Advantage of Nations", ed P. y Janes, Mexico, 1991.

[7] A. Afua, "Innovation management: strategies, implementation, and profits 2003", Oxford University Press, New York, USA, 2003.

[8] M. Porter, "On competition", Harvard Business Press, Boston, MA, USA, 2008.

[9] A. Warren and G. Susman, "Review of innovation practices in small manufacturing companies", Small College of Business, Pennsylvania State University, USA, 2013.

[10] V. Lizardi, F. Baquero and H. Hernandez, "Methodology for a diagnosis on transfer of technology in Mexico", ADIAT, Concyteg editor, Mexico D.F., 2008.

[11] J. Zachman, "A framework for information systems architecture", IBM systems journal, 1987, pp. 276-292 vol 26.

[12] T. Ylimaki and V. Halttunen, "Method engineering in practice: A case of applying the Zachman framework in the context of small enterprise architecture oriented projects", Information Knowledge Systems Management, 2006, pp. 189-209, vol. 5.

 ISBN: 978-1-61208-438-1

[13] J. Sowa and J. Zachman, “Extending and formalizing the framework for information systems architecture”, IBM Systems Journal, 1992, pp. 590-616, vol. 31.

[14] The Open Group, ”TOGAF”,http://www.opengroup.org/togaf/ [retrieved: 05-2015].

[15] S. Bernard, “Enterprise architecture linking strategy, business, and technology”, AutorHouse, Third edition, USA, 2012, pp. 25-31.

[16] F. Ahlemann, E. Stettiner, M. Messerschmidt, and C. Legner, “Strategic Enterprise Architecture Management” , Springer, Germany, 2013, pp. 5-16.

[17] W. Bruls, M. Steenbergen, R. Foorthius, R. Bos, and S. Brinkkemper, “Domain architectures as an instrument to refine enterprise architecture”, Communication of the association for information systems, 2010, pp. 517-540, vol. 27.

[18] Microsip Co, “Microsip manufacturing for SME’s”, http://www.microsip.com, [retrieved: 05-2015].

 ISBN: 978-1-61208-438-1

An Empirical Investigation on the Motivations for the Adoption of Open Source Software

Davide Taibi
Faculty of Computer Science
Free University of Bolzano-Bozen
Bolzano-Bozen, Italy
Email: davide.taibi@unibz.it

Abstract— **Open Source Software has evolved dramatically in the last twenty years and now many open source products are considered similar, or better, than proprietary solutions. The result is that the trustworthiness of some open source products is now very high and the motivations for adopting an open source product over a proprietary product has changed in the last ten years. For this reason, we ran a mixed research approach, composed of three empirical studies, so as to identify the motivations for the adoption of open source products. The goal is to take a snapshot of the state-of-the-art in FLOSS motivation's adoption. Results show that the economical aspects and the freedom of some type of licenses are not the main adoption drivers any more while other motivations such as the ease of customization and ethical reasons are currently considered more important.**

Keywords-Open Source Adoption; Empirical Study; Open Source Quality.

I. INTRODUCTION

Previous research on the adoption of Free/Libre Open Source Software (FLOSS) has mainly focused on adoption models, such as MOSS [1], Open BQR [2], QSOS [3], and others based on the evaluation of a set of information usually considered by potential users when they select a new FLOSS product.

Some works highlight economic or technological reasons [4][5][6] but, to the best of our knowledge, only a few of these studies investigated the factors considered during the adoption of FLOSS by different organization[6][7]. Therefore, the goal of this study is to understand the current reasons that drive the adoption of FLOSS in IT companies, using a research approach that promises to obtain a more complete picture of the motivations for FLOSS adoption. This led to the definition of the following research question:

RQ1: What are the motivation drivers behind the choice of a specific FLOSS product over proprietary software?

In order to answer our research question, we designed a mixed research approach, composed of three empirical studies. We started with a first round of interviews to identify the high level motivations for the adoption. Then, the motivations were refined and clustered by means of a focus group with experts in FLOSS adoption. Finally, we conducted a survey to understand the importance of the motivations identified from the adopter's point of view.

Results of this work show that the motivations for the adoption of FLOSS have evolved in the last years and economical aspects and the license type are not as important as in the past [6] while other motivations, such as the ease of customization and ethical reasons are considered more important.

The rest of this paper is organized as follows. Section II describes the related works. Section III addresses the research approach used. Section IV describes the results obtained. In Section V, we discuss results and in Section IV we present threats to validity. Finally, in Section VI we draw conclusion and future studies.

II. RELATED WORKS

In our previous work [6], we conducted a survey with 151 FLOSS stakeholders, with different roles and responsibilities, about the factors that they consider most important when assessing whether FLOSS is trustworthy. Here, we did not ask the motivations for the adoption of a FLOSS product or a proprietary one, but we asked for the factors considered to compare two FLOSS products.

We identified 37 factors, clustered in five groups: economic, license, development process, product quality, customer-oriented requirements. The product reliability and the degree to which a FLOSS product satisfies functional requirements turned out to be the most important factors for a trustworthy product, immediately followed by interoperability, availability of technical documentation, maintainability, standard compliance and mid-/long-term existence of a user community. Economic factors, such as Return on Investment (ROI) and Total Cost of Ownership (TCO), and the availability of a solid maintainer organization were far from being considered as relevant, as was widely publicized.

Yan et al [9] ran a survey with students as participants, to identify the motivations for the adoption of FLOSS in Malaysia, China, Singapore, Thailand and Vietnam, collecting 264 questionnaires for FLOSS adopters and 212 for non-adopters. They identified a set Intrinsic Motivation (to know, to accomplish, and to experience stimulation), and extrinsic motivations (identified regulation, "introjected regulation" and external regulation).

Previous works discuss some economic motivations [4][10], suggesting TCO and ROI as the most important driver for FLOSS adoption.

Other motivations can be derived by the information required by the FLOSS adoption models. These models are based on the evaluations of a set of information, weighted for their importance. Some models allow users to define the importance of some information, and to evaluate the product

 ISBN: 978-1-61208-438-1

they are willing to adopt. The most important methods in this category are the Open Business Readiness Rating (OpenBRR)[13], the Open Source Maturity Model (OSMM)[12], the Qualification and Selection of Open Source Software (QSOS)[3] and the Open Business Quality Rating (OpenBQR)[2] that summarizes the benefits of the previous three models. All these models, suggest to evaluate economic factors, license, development process, product quality but only QSOS and OpenBQR add customer related factors, such as the degree of which a product satisfies the customer requirement. Other evaluation models are based on a set of predefined weight for each information and allow predicting the trustworthiness or the likelihood of the adoption of a specific FLOSS product. An example is the MOSS model[1], based on the results obtained in [6].

Also in case of the evaluation models, there is an indication on the information that should be evaluated, but the motivations of the choice are not clearly identified.

Finally, a Gartner's report [17] shows that the top three reasons for using FLOSS from manager's point of view are the Total Cost of Ownership (TCO), the improved security and the strategic and competitive advantages.

III. The Research Approach

In this section, we introduce the research approach and the study design of our work.

The goal of this work is to understand the motivations for the adoption of FLOSS software. To avoid to bias the results, based on the results available in the literature, we decided to collect the motivations from scratch by means of a first round of interviews, and then a second run of interviews to analyze qualitative and quantitative results.

This work has been composed by 3 empirical studies, as depicted in Figure 1: 1) Interviews 2) Focus group and 3) Group Interviews.

Figure 1. The Study Process

The first round of interviews has been carried out by means of a questionnaire based on open-ended questions, so as to not drive the interviewee to a predefined set of answers. Then, the focus group has been designed to cluster the answers provided by the participants in smaller sets. Finally, the survey has been conducted by means of a second questionnaire, composed by closed-answer questions, based on the motivations group identified in the focus group.

A. Interviews

The goal of this study was to identify the motivations that influence the adoption of FLOSS.

The interviews are addressed to assessing the current situation in the FLOSS adoption. The idea is to take a snapshot of the state-of-the-art in FLOSS motivation's adoption according to developers, managers and custom integrators. For this reason, we designed a semi-structured interview with open-ended questions.

Semi-structured interviews tend to be much more highly interactive and allow us to clarify questions for respondents and probe unexpected responses. Moreover, in order to collect a set of reliable answers, all interviews were carried out in person, by the same interviewer. We believe this is the most effective way to elicit information and establish an effective communication channel with the interviewees.

The semi-structured questionnaire was composed of three sections. After a brief first section, to profile the interviewee, and the company the interviewee belongs, we asked to list and rank the motivation for the adoption of FLOSS software, based on their importance, on a 0-to-10 scale, where 0 meant "totally irrelevant" and 10 meant "fundamental". During the second section, the interviewer, also took note of the description of the motivation, so as to ease the clustering process to be carried out in the focus group. In the third section of the questionnaire, since we wanted to understand if the factors identified in [6] influence the adoption of FLOSS, the interviewer asked to rank the factors identified in [6], not listed as motivations during the second section of the questionnaire.

B. Focus Group

The focus group has been carried out to discuss the results of the interviews and draw qualitative conclusions on the results, summarizing and clustering the motivations. The clustering part is needed, since several users can define similar but not identical motivations.

The focus group event has been planned to last three hours. We invited five participants; four researchers with experience in FLOSS quality, adoption models and FLOSS development and the author of the paper that acted as moderator.

During the focus group we did not report the importance of each motivation to the participants, so as to avoid biased results by this value.

Before the beginning of the focus group, we provided an overview of the objectives of the study, and described how participants should discuss and act during the session. Then, we presented the motivations elicited in the survey and we wrote them on a set of post-it notes.

In order to better understand the difference among similar motivations, the moderator, who also carried out the interviews in person, reported the description of the motivation reported by the interviewees. Then, we asked the participant to organize the post-it notes on a white board using the affinity grouping technique[16].

C. Group Interviews

The final study has been designed to be executed in a group interview, with the support of a closed-ended questionnaire.

In this case, the interviewer explained each question to the participants who answered to the questions on a paper-based questionnaire.

The interviewer distributed the questionnaires to the participants before the beginning of the workshop and then,

 ISBN: 978-1-61208-438-1

after a short introduction of the motivation of the study, he asked to fill in the questionnaire, taking care that participants were not influenced in their answers from each other's.

We believe this method is more effective than online questionnaires, since participants have the possibility to make questions or to ask for more details.

We organized the questions in the questionnaire in two sections, according to the types of information we sought to collect:

- *Personal information, and role in relation to FLOSS*: helps to profile the interviewee and the company.
- *Motivations:* here we asked to list and rank the motivation for the adoption of FLOSS software, based on their importance, on a 0-to-10 scale, where 0 meant "totally irrelevant" and 10 meant "fundamental".

The motivations included both the motivation identified in the interviews and the missing factors identified in [6]. Moreover, to improve the readability of the questionnaire, we grouped the motivations in five groups: License, Development process, Product quality issues, Customer requirements.

IV. RESULTS

Here we report the results of the three studies, together with a short discussion and interpretation.

In order to answer to our research question, we first analyze the results of the group interviews and then we compare the results with those obtained in the first interviews after the clustering carried out in the focus group.

Finally, we compare the list of motivations with the factors highlighted in our previous survey [6].

A. Interviews

The sample of interviewees was not determined in advance. A preplanned sample would have allowed for a more controlled result analysis, but it would also have limited the possibility to add interviewees to the set in an unanticipated manner. We are fully aware that this may have somewhat influenced our results.

Here we first provide information about the sample of respondents, which can be used to better interpret the results and then, we show the collected results with a concise analysis of the responses obtained, with insights gained by statistical analysis.

TABLE I. INTERVIEWEER COMPANY SIZE

Company Size	Percentage
SMEs (<250 employees)	47.5%
Medium Enterprise (250-500 employees)	17.5%
Industry (>500 employees)	35.0%

Table I contains the distribution of companies where our interviewees belong, while Table II show the percentages of the roles for four organizational roles identified in the questionnaire. Note that roles may not necessarily be mutually exclusive.

TABLE II. INTERVIEWEES ROLES

Role or title	Percentage
Manager	35%
Developer	27.5%
Custom integrator	52.5%
End user	20%

TABLE III. THE MOTIVATIONS OBTAINED FROM THE INTERVIEWS

Reason	#Answers	Freqency %
Ethic	13	34.21
Customization Easiness	9	23.68
Personal Enrichment	4	10,53
Synergy	6	15.79
Quality	6	15.79
Economic	7	18.42
Community support	5	13.16
Support	6	15.79
Flexibility	6	15.79
Free	9	23.68
Innovation	4	10.53
Documentation	3	7.89
Works better than CSS	6	15.79
Personal Productivity	3	7.89
Company decision	4	10.53
Maturity	4	10.53
Better Solution than CSS	4	10.53
Adaptability	5	13.16
Competitiveness	6	15.79
Community Enrichment	3	7.89
Avantgarde	1	2.63
Free Availability	1	2.63
Completeness	1	2.63
Customer requirement	1	2.63
Customer need	1	2.63
Economic model	2	5.26
Fast Evolution	1	2.63
Freedom	1	2.63
Internal Management	1	2.63
Independence from other SW	1	2.63
Platform Independence	2	5.26
Long-term investment	1	2.63
License Cost	2	5.26
Reduced investment for clients	1	2.63
Partnership	2	5.26
Professional Support	1	2.63
Reuse	1	2.63
Standards	1	2.63
Multiplatform development	1	2.63
Vendor lock-in	2	5.26
Training	1	2.63
Free updates	2	5.26
Higher consultancy value	1	2.63
Trustworthy	4	10.53

 ISBN: 978-1-61208-438-1

We interviewed a total of 38 participants. collecting 52 different motivations with an average of 4.32 motivations listed per interviewee, a minimum of 3 and a maximum of 12 motivations. After a first screening on synonyms (eg. "Ethic" and "Ethical reasons") we reduced the total number of motivations to 33.

In Table III, we report the list of reasons together with their frequencies, ordered by #answers. Column Frequency % reports the answer's frequency (#Answers/Total number of Answers).

The first immediate result is that, compared to our previous survey [6], several new motivations have been identified while others are not considered. Unexpectedly, most of the development factors, license issues, and quality aspects such as complexity, performance and usability are not considered as good motivations for FLOSS adoption.

However, since several motivations identified in this first round of interviews are pretty similar, the identification of similarities and difference will be analyzed after the results of the focus group.

B. Focus Group

During the focus group we discussed how to cluster similar motivations, and how to compare them to those highlighted in our previous survey [6].

The clustering, carried out with the affinity group technique, allowed to reduce the motivations to 21, on which 13 are common with the factors identified in [6] and 8 are new: ethic, personal productivity, freedom, partnership, competitiveness, innovation, flexibility, project maturity.

Table VI shows the list of motivations after the clustering process carried out, together with the results obtained in the next study. For reason of space, Table IV do not report the motivations not considered by our interviewees.

Based on the clustering results, we were also able to calculate the ranking of each motivation reported in the interviews. For reason of space, we do not report the results but we only describe the differences with the results obtained in the group interviews, in the next section.

C. Final Group Interviews

As for the interviews, the sample of interviewees was not determined in advance.

The survey has been executed during FOSDEM 2013 workshop "An Interactive Survey on marketing and communication strategies"[14]. We distributed 47 questionnaires, obtaining 21 valid questionnaires.

Participants were FLOSS experts, developers and practitioners. No students or non-experienced participants were considered in the analysis of the results. Table IV contains the distribution of companies where our interviewees belong while Table V shows the percentages of the roles for four organizational roles. Note that, in this case, roles are mutually exclusive, since we asked our interviewees to answer to the questions based on the selected position.

Even if the sample was not determined in advance, the roles of our interviewees are well distributed among managers, developers and custom integrators, while no end-users filled in the questionnaire in this group.

TABLE IV. Company Size

Company Size	Percentage
SMEs (<250 employees)	33.3%
Medium Enterprise (250-500 employees)	42.8%
Industry (>500 employees)	23.9%

TABLE V. ROLE

Role or title	Percentage
Manager	9.5%
Developer	52.4%
Custom integrator	38.1%
End user	0%

A statistical analysis of the responses lets us partition the factors into importance groups, which we show in Table VI's columns "entire dataset," "managers", "developers" and "customer integrators".

As for Table III, the column "Frequency %" reports the answer's frequency (#Answers/Total number of Answers) while column "rank" report the weighted average of ranking, using the importance of each motivation as the weight while column "[6]" shows if the motivation has an higher or lower importance than the relative factor identified in [6].

Let's first discuss the column "entire dataset, where we identified eight importance groups, from 1 (least important) to 8 (most important). The ordered grouping indicates a statistically significant importance ranking between motivations belonging to different groups, but no such ordering within each group. For instance, the motivation "customization easiness" belongs to group 8, so it's ranked as more important than quality and just as important as "Economic" and "Personal Productivity" which are in group 6. The number of groups depends on the portion of the population considered. For "Managers", the statistical analysis led to nine groups while for "Developers and "Custom Integrators" to eight groups.

The motivation "customization easiness" is considered, by all groups, as the most important driver for the adoption of FLOSS. Compared to our previous survey [6], this motivation gained several positions, moving from group 4 (out of 8) to group 8.

Ethical motivations, not included in [6], seems to be very important for our interviewees while the overall product quality is at the same level of personal productivity, and economic.

Other motivations such as freedom, community support and potential partnership are relatively important (group 4) while all other motivation are definitely not relevant, lying in groups 1, 2 or 3.

 ISBN: 978-1-61208-438-1

TABLE VI. FINAL SET OF MOTIVATIONS (GROUP INTERVIEWS)

Reason	Entire Dataset			Managers			Developers			Custom Integrators		
	Rank	*freq%*	*[6]*	*Rank*	*freq%*	*[6]*	*Rank*	*freq%*	*[6]*	*Rank*	*freq%*	*[6]*
Customization Easiness	8	61.90	↑	7	0.14	↑	8	0.78	↑	8	1.00	↑
Ethic	7	66.67		7	0.29		8	0.78		7	1.00	
Quality	6	71.43	↓	6	0.43	↓	6	1.00	↓	8	0.60	↓
Personal Productivity	6	47.62		1	0.29		7	0.89		7	0.60	
Economic	6	52.38	↑	9	1.00	↑	2	0.56	↓	4	0.60	↓
Freedom	4	38.10					5	0.67		3	0.40	
Support (community)	4	42.86	↑	2	0.29	↓	3	0.89	↓	6	0.80	↑
Partnership	5	33.33		7	0.57					4	0.80	
Competitiveness	2	28.57		3	0.71					2	0.80	
Security	2	19.05	↓				2	0.33	↓	3	0.80	↓
Innovation	2	23.81		3	0.43					2	0.20	
Multiplatform devel.	2	9.52	↓	2	0.43	↓	3	0.22	↓			
Flexibility	2	23.81								2	0.60	
Imposed by the company	1	14.29	=	1	0.29	=	1	0.22	=			
Maturity	1	14.29					1	0.11		2	0.60	
Reliability	1	9.52	↓	1	0.21	↓				1	0.60	
No Vendor Lock-in	1	9.52	=	1	0.14	=	2	0.11				
Customer Requirement	1	4.76	=	1	0.14	=						
Free Updates	1	4.76	↓	1	0.29	↓						
Training	1	0.00		0	0.00							
Reuse	1	0.22	↓	0	0.00		2	0.22	↓			

When considering the different roles. few noticeable differences emerge.

As expected, managers consider Economic of higher, but developers and Custom Integrators consider its importance substantially below average. Unexpectedly, managers have a different view of Ethic and personal productivity considering both motivations of little importance compared to the other groups. Moreover, managers are not interested to freedom at all.

Custom integrators consider only a smaller set of motivations compared to the other groups, with quality, customization easiness, ethic and personal productivity as the most important motivations.

Developers' motivations are similar to the average.

Merging the result of the first set of interviews with those obtained in the group interviews, we will obtain a final dataset composed by 59 participants (38 from the one-to-one interviews and 21 from the group-interviews).

In this case, results do not change significantly, showing a similar trend as in Table VI. For reason of space, we report only the variations respect to the results presented in Table VI.

The only differences are in three motivations where Ethic move down to group 5, personal productivity moves up to group 6 and community support moves up to group 5.

We believe that this is due to the population of the interviewees, that in the interviews mainly belong to medium enterprises while in the group interviews to SMEs.

V. RESULTS DISCUSSION

The first immediate result of the study is that several development, economical and quality factors, usually considered important to evaluate a FLOSS product [6] are not considered as good motivations that drive the adoption of FLOSS.

Our interviewees prefer FLOSS since they can easily customize it, without dealing with proprietary issues and being able to provide the higher value as possible to their customers.

Ethical motivations gained a very high importance. We believe that this is due to the population of our interviewees, since we carried out the interviews during FOSDEM.

As expected, quality and economic are always considered very important while new motivations as personal productivity. Freedom, potential partnerships are also considered as adoption key drivers.

VI. THREATS TO VALIDITY

Due to the number of subjects we were able to obtain necessary power for performed statistical tests. Before performing test preconditions (normality, independence of variables, etc.) were checked to make sure that they are satisfied.

To get reliable measures questionnaires were checked by an expert on empirical studies.

Subjects have similar background and knowledge about FLOSS.

Although we ask the participant of the survey to provide individual answers, the results could be partially affected

 ISBN: 978-1-61208-438-1

since they were seated together in the same room. In the first two studies we employed only high skilled participants with a good experience on FLOSS while in the survey, we only analyzed the answers provided by experts. However, since we ran the survey during a FLOSS conference results could be biased in favor of FLOSS.

VII. CONCLUSION

In this work, we reported on a mixed research approach composed of three empirical studies, with the overall goal to characterize the motivations for the adoption of FLOSS products.

We first provided an overview on the existing proposal and studies investigating the motivations, including our previous survey [6] where we analyzed the factors considered by the users when they need to compare two or more FLOSS products.

Then, we conducted a first round of semi-structured interviews of 38 FLOSS users, so as to identify the high level motivations and to understand if the factors identified in [6] can also be considered motivations. Results of this first study show that most of the quality and economic factors are not driving the choice of FLOSS among proprietary software. The motivations were then clustered in groups, reducing the set to 21 motivations.

Finally, we conducted a structured group interview, based on a closed-answer questionnaire, where we asked our interviewees to rank the motivations they consider key drivers for the FLOSS adoption.

Results show that FLOSS users currently consider new motivations. Ease of customization is the most important motivator to adoption of a FLOSS product since it allows companies to better adapt the product to their customers.

Ethics is also a very important motivation. Several users consider it more ethical to adopt FLOSS instead of proprietary software.

Finally, quality, economic and personal productivity are also considered of middle importance with some variations in different groups. For instance, managers are most interested in economics, with less emphasis on ethics, as also confirmed by Gartner in [17], while customer integrators consider product quality as the most important motivation.

Although we designed both studies to minimize threats to validity, it was difficult to obtain good statistical significance for each group of users. We plan to replicate the study with a larger set of users so as to validate the results and to improve its statistical significance.

REFERENCES

[1] V. Del Bianco, L. Lavazza, S. Morasca, D. Taibi, and D. Tosi, "Quality of Open Source Software: The QualiPSo Trustworthiness Model." In: proc. 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skövde, Sweden, pp.199-212, June 3-6, 2009

[2] D.Taibi, L.Lavazza, and S. Morasca, "OpenBQR: a framework for the assessment of OSS." Open Source Development, Adoption and Innovation. pp.173-186, 2007.

[3] Atos Origin, "Method for Qualification and Selection of Open Source software (QSOS), version 1.6", http://www.qsos.org/download/qsos-1.6-en.pdf

[4] B. Buffett, "Factors influencing open source software adoption in public sector national and international statistical organisations." Meeting on the Management of Statistical Information Systems (MSIS 2014), Dublin, Ireland and Manila, Philippines

[5] V. Del Bianco, L. Lavazza, V. Lenarduzzi, S. Morasca, D. Taibi, and D. Tosi, "A study on OSS marketing and communication strategies" 8th IFIP International Conference on Open Source Software, OSS 2012, Hammamet. vol. 378, pp. 338-343, 2012.

[6] V. Del Bianco, L. Lavazza, S. Morasca, and D. Taibi, "A Survey on Open Source Software Trustworthiness." Software, IEEE. Vol.28, pp.67-75, 2011.

[7] R. Dirk, "The economic motivation of open source software: Stakeholder perspectives." Computer. Vol. 40(4), pp.25–32, April 2007

[8] V. Del Bianco, L. Lavazza, S. Morasca, and D. Taibi, "The QualiSPo approach to OSS product quality evaluation." Workshop on Emerging Trends in FLOSS Research and Development (FLOSS-3). Cape Town, South Africa pp.23-28, 2010 .

[9] Y. Li, C. H. Tan, H. Xu, and H. H. Teo, "Open source software adoption: motivations of adopters and amotivations of non-adopters." ACM SIGMIS Database. Vol.42, pp.76-94, 2011.

[10] V. Lenarduzzi, "Towards a marketing strategy for open source software" Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement. Pp. 31-33, 2011.

[11] V. Kumar, B. R. Gordon, and K. Srinivasan," Competitive Strategy for Open Source Software. Marketing Science. Vol. 30, pp1066-1078, 2011.

[12] B.Golden "Making Open Source Ready for the Enterprise: The Open Source Maturity Model", from "Succeeding with Open Source", AddisonWesley, 2005

[13] "Business Readiness Rating for Open Source - A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software", BRR 2005 RFC 1, http://www.openbrr.org.

[14] E. Petrinja, R. Nambakam, and A. Sillitti, "Introducing the OpenSource Maturity Model." In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS '09). IEEE Computer Society, Washington, DC, USA.Pp.37-41, 2009.

[15] D.Taibi, and V.Lenarduzzi, "An Interactive Survey on marketing and communication strategies" FOSDEM'13. Bruxelles, February 2013.

[16] S.Mizuno, "Seven New Tools for QC: For Managers and Staff Promoting Company-wide Quality Control", 1979. Mikka Giren

[17] M.Cheung, and L. F. Wurster, "Open-Source Software Adoption and Governance, Worldwide, 2014" Gartner Report feb. 2015. G00272505

 ISBN: 978-1-61208-438-1

Gamifying and Conveying Software Engineering Concepts for Secondary Education: An Edutainment Approach

Roy Oberhauser
Computer Science Dept.
Aalen University
Aalen, Germany
roy.oberhauser@hs-aalen.de

Abstract—**Because of its abstract nature, software engineering faces image, perception, and awareness challenges that affect its ability to attract sufficient secondary school age students to its higher education programs. This paper presents an edutainment approach called Software Engineering for Secondary Education (SWE4SE), which combines short informational videos and a variety of simple digital games to convey certain SE concepts in an entertaining way. Our realization mapped various SE concepts to seven digital games, and results from an evaluation with secondary students showed that a significant improvement in the perception, attractiveness, and understanding of SE can be achieved within just an hour of play. Thus, we suggest that such an edutainment approach is economical, effective, and efficient for reaching and presenting SE within secondary school classrooms.**

Keywords-software engineering education; software engineering games; game-based learning; digital games.

I. INTRODUCTION

The demand for software engineers appears insatiable, and computer science (CS) faculties and the software engineering (SE) discipline appear to be steadily challenged in attracting and supplying sufficient students to fulfill the demand. While it may appear to each higher education institution and country to be a local or regional problem, the challenge may indeed be more common and broader in nature. For example, in the United States in 2005, after a boom beginning around 2000, a 60% decrease over 4 years in the number of freshmen specifying CS as a major was observed [1]. And US bachelor degrees in CS in 2011 were roughly equivalent to that seen in 1986 both in total number (~42,000) and as a percentage of 23 year olds (~1%) [2]. As another example, Germany also observed a negative trend from a record number of CS students in 2000, and one 2009 news article [3] blamed the negative image of CS among the young generation. While the number of starting CS undergrads in Germany has since increased, roughly 33,000 software developer positions remain unfilled in 2014 [4]. In addition, the forecast demographic effects in certain industrial countries imply a smaller younger population available to attract, reducing the overall supply and thus increasing the competition between disciplines to attract students and workers. It is thus a critical and continual worldwide issue to attract young women and men to SE.

Concerning SE's image, according to D. Parnas [5] there is apparently confusion in higher education institutions as to the difference between CS and SE, and we assert this affects secondary education as well. The CS equivalent term in Germany, *informatics*, is much more publically known and marketed as a discipline or field of study than is SE. Thus, SE programs struggle in the overall competition between disciplines to attract secondary students for this critical area to society, since SE must first raise awareness about its field.

The concepts inherent in SE, as exemplified in the SWEBOK [6], tend themselves to be abstract and to deal with abstractions. Thus, they are difficult to convey, especially to secondary school students who have not significantly dealt with such abstractions, and cannot thus practically imagine what they mean. Furthermore, secondary school teachers and institutions are challenged in teaching CS, and have likely themselves not been introduced to SE.

Learning is a fundamental motivation for all game-playing, as game designer C. Crawford [7] once stated. With this in mind, this paper contributes an SE edutainment approach we call SWE4SE for gamifying and conveying SE concepts with secondary school students as the target audience. It describes the principles in the solution concept and example mappings of SWEBOK concepts onto relatively simple digital games, demonstrating its viability, and the evaluation with secondary students showed that combining short informational videos and a variety of simple digital games can be economical, effective, and efficient for improving SE awareness, perception, and attractiveness.

The paper is structured as follows: Section II describes related work. Section III describes the SWE4SE solution principles, our game design, and the incorporated SE concepts. Section IV describes our realization. Section V details the evaluation, followed by a conclusion.

II. RELATED WORK

Serious games [8] have an explicit educational focus and tend to simulate real-world situations with intended audiences beyond secondary education. [9] performed a literature search of games-based learning in SE and "found a significant dearth of empirical research to support this approach." They examine issues in teaching the abstract and complex domain of requirements collection and analysis and, more generally, SE. Using a constructivism paradigm, the role-playing client-server-based SDSim game has a team manage and deliver multiple SE projects. The systematic

 ISBN: 978-1-61208-438-1

survey on SE games by [10] analyzed 36 papers, all of which had targeted undergraduate or graduates. Regarding secondary education, whereas initiatives for teaching programming are more common, conveying SE concepts in general and gamifying SE has not hitherto been extensively studied, nor has the educational value of explicitly "non-serious" (or fun) games for this population stratum.

Concerning the perception and attractiveness of CS among secondary students, the study by [11] of 836 High School students from 9 schools in 2 US states concluded that the vast majority of High School students had no idea what CS is or what CS majors learn. This conclusion can most likely be transposed to the lesser known discipline of SE.

In contrast, SWE4SE is targeted not towards higher education, but rather secondary school students with an explicit non-serious game approach. Our results compare with [11], but go further in showing that an edutainment approach can improve the perception and attractiveness of SE. Compared to other learning game approaches, it explicitly makes the tradeoff to value entertainment more and education less in order to retain student engagement and enjoyment. It also explicitly includes short informational and entertaining video sequences to enhance the experience beyond gaming alone.

III. SOLUTION

SWE4SE consists of a hybrid mix of short informational and entertaining videos and a variety of relatively simple digital games. Our solution is necessarily based on certain assumptions and constraints. For instance, we assumed that the players may not only be playing in a compulsory classroom setting, but may play voluntarily on their own time, meaning that they could choose to stop playing if it becomes boring or frustrating and discard the game for some more interesting game. Thus, the edutainment is considered to be "competing" with available pure entertainment options. However, we expect that the game may be promoted to secondary school teachers where they would introduce students to the game, meaning that our concept must not necessarily compete solely with commercial products and other entertainment. We also assumed that the motivational factors for students in the SWE4SE are curiosity, exploration, discovering different games, and finding fun areas.

A. Solution Design Principles

Web-browser Principle (P:Web): To broadly reach the target audience (secondary students ages 12-18), we chose to focus our initial design and implementation on a web-based game solution and avoid the installation of components on game or various OS platforms. This constrains the available game options, but increases the reachable population.

Engagement / Enjoyment Principle (P:En): We want to keep the students engaged and to emotionally enjoy the experience. To reduce the likelihood of a player discontinuing due to lack of fun, we chose to value and prioritize the fun aspect more than pushing the learning of SE educational concepts. We are thus aware of the fact that less of the SE material may be conveyed and retained, but by retaining engagement over a longer timeframe, further possibilities for SE concept conveyance result.

Game Reuse Principle (P:GR): Leverage known games and game concepts (repurposing) when possible, such as those in [12]. Players may thus already know the basics of how the original game works - reducing the time to become proficient, and they may find the new twist involving SE concepts interesting. Also, more time and cognitive capacity may be available for the mapping of SE concepts to the game elements when compared with completely unfamiliar games.

Simple Game Principle (P:SG): Utilize relatively simple games when not utilizing already known games (cp. *P:GR*). This reduces the overall effort required to acquire game proficiency and to acquire the SE concepts.

SE Concept Reinforcement via Game Action Principle (P:GA): during the games, immediate feedback messages that reinforce game associations to SE concepts are given, e.g., "Correct, the quality was OK" or "Oops, the component was released with quality defects" for a software quality assurance (SQA) game. This makes it more transparent how choices and actions are reflected in SE concepts.

B. Edutainment Elements and SE Concept Mappings

We believe that certain aspects of SE cannot be conveyed well solely with games and should thus be supplemented.

Text components: a brief amount of onscreen text was used to introduce the topic area, relevant SE concepts, and the game objective and major game elements. Such a short text that can be clicked away does not overly interfere with the experience, and can be read or skimmed rather quickly. Using these, later bonus-level text questions can reference some prior text or video as a way to verify comprehension.

Video components: a short 5-minute informational video described how prevalent code is, society's dependence on software, and how important software development and software engineers are. The ability to include relevant videos, and the ability for users to explore and discover such videos, adds to the "adventure".

Game components: Various concepts from SWEBOK were chosen, with the selection subjectively constrained by our project resources, technical and game engine constraints, and how well a concept might map to a game concept. The selection, mapping, and prioritization of what to realize was subjectively reflected and decided on as a team, which is summarized in Table I.

TABLE I. SE CONCEPT TO GAME MAPPING

SE Concept	Analogous Common Game	SWE4SE Game Variant
1) Processes	Pac-Man	ProcMan
2) Quality assurance	Pinball	Q-Check
3) Requirements	Tower Defense	ReqAbdeck
4) Testing	Angry birds	Angry Nerds
5) Construction	Angry birds	Reverse Angry Nerds
6) Defect remediation	Space invaders	Bug Invaders
7) Project management	Maze	Path Management

The mapping should be interpreted as exploratory and not prescriptive; our intention here is rather to demonstrate the possibilities available in such an edutainment approach.

 ISBN: 978-1-61208-438-1

IV. REALIZATION

Scirra Construct 2 was used to develop the games. Layouts and Event sheets were used, and each game object type is given properties with certain states and behavior. Sounds and music were integrated. The web browser requires HTML5 and Javascript support. Text components were initially German, but could be readily internationalized. For brevity, details on how points are earned, bonus levels, speed changes, or lives lost in each game are omitted.

A. Conveying SE Concepts in the Various Games

For each SE concept below, how the analogous common game was mapped to corresponding primary game goals is described and depicted.

1) SE Processes: to convey an engineering process, we chose to introduce sequential activities common to many engineering processes. Based on the waterfall process, these were Analysis, Design, Implementation, Testing, and Operations (ADITO, or equivalently AEITB in German). We also provided a test-driven development (TDD) variant where the Testing occurs before Implementation (ADTIO).

ProcMan: this game is analogous to the well-known Pac-Man game (see Figure 1), with a twist that, whereas in PacMan one got points by traveling everywhere in the maze in any order, the goal here for the player is to follow a given SE process sequence by making ProcMan consume the distributed letters in the given order while avoiding ghosts.

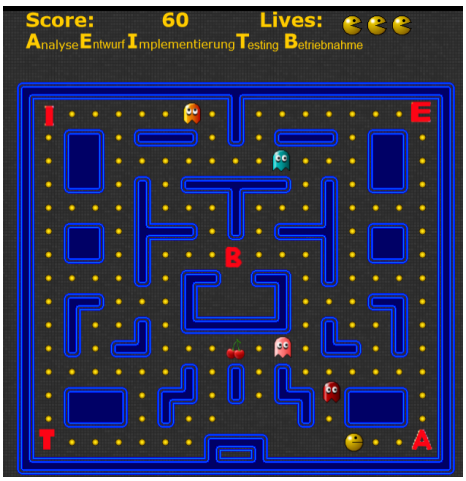


Figure 1. ProcMan game conveys SE processes (screenshot).

2) Software quality assurance: SQA differs on the type of software component being inspected (e.g., GUI, database, business logic). Quality awareness and attention to detail matter, and the appropriate QA tools and testing procedures must be chosen.

Q-Check: this game is loosely analogous to pinball (see Figure 2). Software components (SoCos) portrayed as colored shapes spin and drop into a funnel, while a cannon (blue on the left) automatically shoots them individually after a certain time transpires (indicated via a decreasing green bar on the cannon). The player's goal is to select the process (tunnel on the right) that matches the SoCo type currently in the cannon based on both color and shape, or reject it for rework (yellow) if it is defective, improving the future error rate.

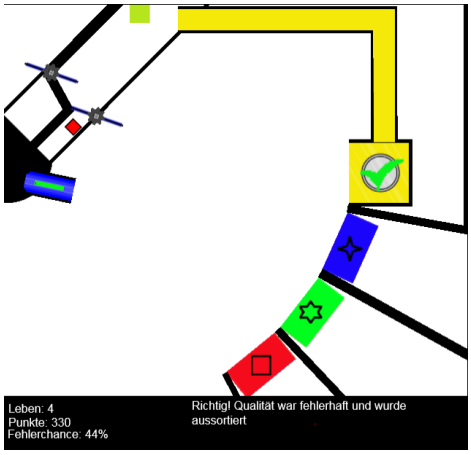


Figure 2. Q-Check game conveys SE quality assurance (screenshot).

3) Software requirements: this concerns itself with the SE concept of requirements coverage, for instance not overlooking a requirement, determining which requirements to fulfill how and when (different requirement types need different specialized competencies), which requirements to jettison (e.g., due to insufficient business value).

ReqAbdeck: (*Abdeckung* in German means "coverage") this game is analogous to the popular game Tower Defense (see Figure 3), whereby here waves of "reqs" (requirements as colored triangles) flow from left to right, and towers in various colors that cover (fire) only at their requirement color must be dragged to the appropriate vicinity before the "reqs" reach the gate. The towers disappear after a short time indicated on their border. Thus, one is not covering critical requirements in time with the matching implementation, ignoring or forgetting a requirement, or not dropping via a gate those requirements without business value (denoted by black circles). One example action message here is "Great, requirement was covered by a suitable realization.

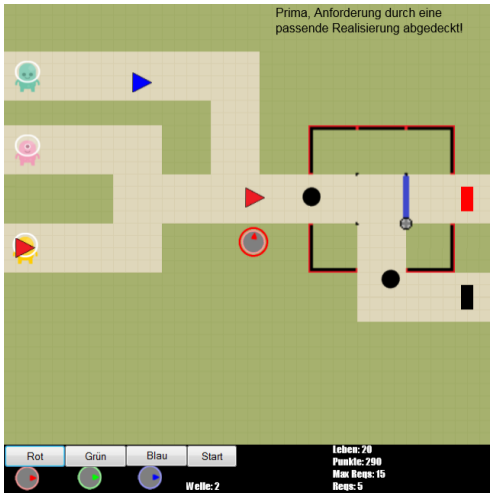


Figure 3. ReqAbdeck conveys SE requirement coverage (screenshot).

4) Software testing: the focus here is determining where to test to find deficiencies in some software construct.

Angry Nerds: this game is loosely analogous to the popular game Angry Birds (see Figure 4). For various reasons we chose to depict hardware-like testing here of children's blocks, since it was not obvious to us how to convey code-based testing in an obvious manner without necessitating extra explanations. The player's goal in this case is to test a given construct of slabs surrounding PCs by determining where and how hard to throw a mouse at it to knock it completely over. They realize that multiple tests are necessary to discover its weaknesses.

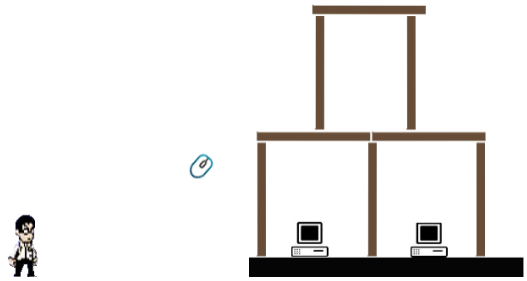


Figure 4. AngryNerds conveys SE testing (screenshot).

5) Software construction: the point here is to engineer or build the software such that it exhibits resiliency.

Reverse Angry Nerds: this is a bonus level of the previous game, and reverses the role as shown in see Figure 5, having the player now try to build a resilient construct by dragging and placing slabs in such a way that it withstands the automated testing (cannonball shot at it).

Figure 5. Reverse AngryNerds game conveys SE construction (screenshot).

6) Software defect remediation: the learning focus is that different defect types require different remediation techniques and countermeasures applied accurately.

Bug Invaders: in this game, analogous to space invaders, a matching remediation technique (ammunition color in the lower shooter) and firing accuracy are needed to destroy exactly that specific bug type that drops down before it creates project damage (see Figure 6).

7) Software project management: here we convey that multiple choices towards optimizing project costs exist. With appropriate planning, the project goal can be reached with the allotted resources, while unexpected problems can be overcome but cost unplanned resources.

Path Management: in this game a player must manage a starting budget in points efficiently (see Figure 7). From the project start (green triangle) a path selection is made to take it to the end (red circle). Red blocks depict possible steps, blue steps the currently available choices, and green the current position. Each step costs 100 points, while randomly generated problems (black circles) add to the planned costs.

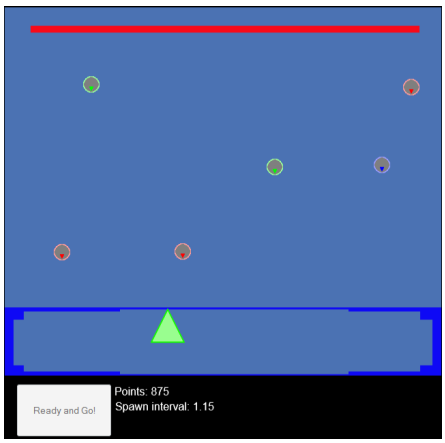


Figure 6. Bug Invaders convey SE defect remediation (screenshot).

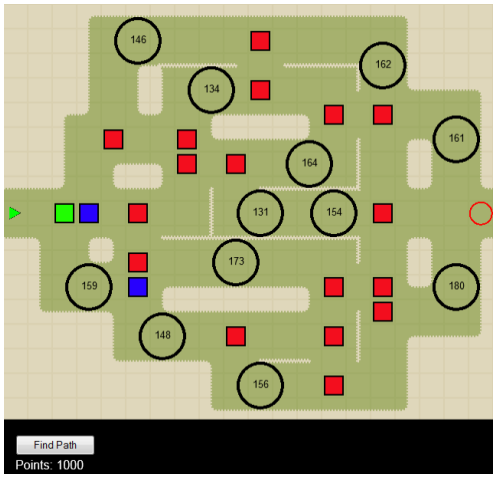


Figure 7. Path Management conveys SE project management (screenshot).

B. Realization of SE Exploration Concept

To tie it all together, the realization includes a SE universe to navigate to and discover various SE planets. Figure 8 shows the spaceship navigating in two dimensions. A shield level, reduced when colliding with asteroids, is shown as a colored line next to the ship. The game ends when the shields are lost or on collision with the sun. The bottom right of the screen shows a navigation map with the location of all planets (red first, green when visited, and violet for the home planet, and the spaceship as an arrow.

When arriving at a planet (Figure 9), a short text about SE concepts that relates to the game is shown, which when understood, can later be used to answer bonus questions at a gate. The portal to the game is shown on the left. The brown gate and fence shows a darkened advanced level area only accessible by successfully passing a gate requiring that SE challenge questions be answered correctly. This then enables passage and undarkens the upper bonus region top.

 ISBN: 978-1-61208-438-1

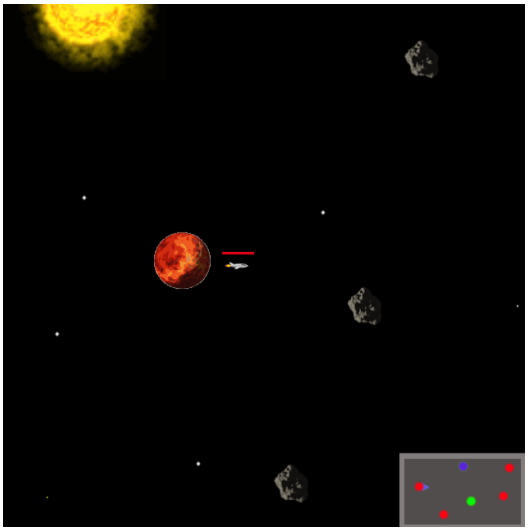


Figure 8. Spaceship navigating the SE universe (screenshot).

Figure 9. Example of a uniquely named SE game planet (screenshot).

On the home planet, a TV tower shows the video.

The realization is economical in that it can be widely distributed (*P:Web*) without client installation costs or large cloud server investments (it runs locally in the browser).

V. EVALUATION

The convenience sampling technique [13], common in educational settings, was selected to evaluate our SE edutainment approach due to various constraints otherwise inhibiting direct access to a larger random population sample of the target population stratum. These constraints include logistical and marketing effort and budget, privacy issues, and acquiring parental consent for school age children.

Setting: Two teachers gave us access to 20 students in informatics interest groups for 90 minutes at two different public university preparatory (secondary) schools in the local region. *Setting A* using an alpha version of the software tested with a group of 8 males, and a later *setting B* using a beta version in a different city with 6 females and 6 males students. Figure 10 shows the age and gender distribution, and Figure 11 indicates their current game usage frequency.

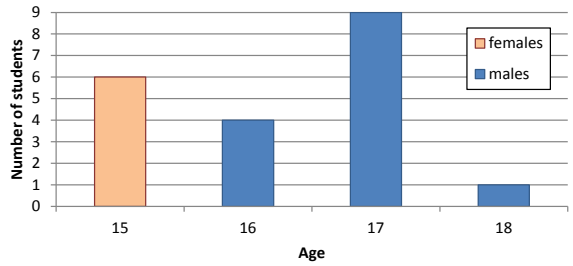


Figure 10. Student age and gender distribution.

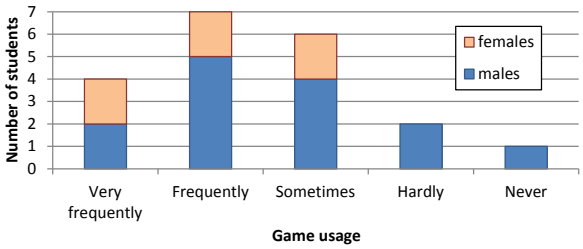


Figure 11. Prior game usage frequency distribution.

Questionnaire: While we considered utilizing the GEQ [14], it appeared more appropriate for more immersive games. Hence, due to the player ages, the limited time they had for playing multiple different short games (7 games in one hour), and the limited time, attention, and incentives for filling out pre- and post-questionnaires (10 minutes respectively), only a few questions about their state before and after with regard to negative or positive affect were included. They were asked but not compelled to answer all questions, so some fields were left blank by some students, which could be interpreted to mean they did not understand the question, did not know how to answer, or did not care to answer the question. Blank answers were thus omitted.

Session: The empirical evaluation consisted of 90-minute sessions held in two different settings A and B. The first 5 minutes consisted of a short introduction as to the purpose of our visit and what to expect, involving no teaching. Students were then given 10 minutes to fill out anonymous printed questionnaires in German that provided us with initial basic data. When all were done, they began their one-hour edutainment experience. In the 10 minutes directly thereafter, monitors were turned off and they completed the second part of their questionnaire, which focused on their experience and understanding, after which we held a 5-minute group feedback session.

Results: We observed that all students were engaged with the software for the entire hour and appeared to enjoy the experience *(P:En),* and occasionally interacted excitedly with fellow students. Below is our analysis of the questionnaire results. Unless otherwise indicated, averages were based on a scale of 1 to 5 (1 being very good, 5 bad):

- Overall experience: 2.1 (good); relates to *P:En*
- Game enjoyment: 2.0 (good); relates to *P:En*
- Helpful conveying several SE concepts via different games: yes (17), no (1); relates to *P:SG* and *P:GR*

 ISBN: 978-1-61208-438-1

- Success rate in correctly recalling the SE concepts associated with each named game (open answers): 62%; relates to *P:GA* and Text components. Note that the game names in the questions could serve as a hint, but these did not include the complete and explicit SE concept nor was the game accessible.

Video:

- Watched the video attentively: yes (20)
- Video and its quality (good/bad): good (20)
- Video length 5 minutes: keep (19), lengthen (1)

Table II shows the change in perception, attractiveness, and understanding of SE after the experience.

TABLE II. CHANGE IN SE PERCEPTIONS

Change in responses	Before	After	Improvement
Importance of SE for society[a]	1.7	1.2	33%
Attractiveness of SE as a personal career path[b]	3.3	2.7	16%
Ability to define what SE is[c]	2.9	2.3	20%

a. Scale of 1 to 3 (1=very important, 3=not important); 2 wrote "don't know" in the prequestionnaire.
b. Scale of 1 to 5 (1=very attractive, 5=not attractive)
c. Answer graded (1 excellent, 2 very good, 3 satisfactory, 4 sufficient) for B group only.

As to interpreting the results, a convenience sample can obviously contain a number of biases, including under- or overrepresentation. Our supervision of the evaluation process and physically observing that the software was actually used for an hour by each of the students separately, and that each questionnaire was individually filled out, removed certain other kinds of threats to validity.

The evaluation showed the effectiveness of the approach: because students in this age group had previous familiarity with gaming, they had no difficulty rapidly discovering and playing the games intuitively without training or help, they understood the intended mapping of SE concepts onto game elements, and the perception, attractiveness, and understanding of SE improved significantly without discernable gender differences. It was efficient in achieving these effects in the relatively short span of an hour of usage.

In summary, an edutainment approach with short videos, short text components, and a variety of simple games appears promising for effectively and efficiently improving the awareness about and image SE, at least for our target population stratum.

VI. CONCLUSION AND FUTURE WORK

We described SWE4SE, an edutainment approach for gamifying and conveying software engineering concepts to secondary school students. Various principles used of the edutainment approach were elucidated, and it was shown how various SE concepts could be mapped and realized with various digital game concepts and elements. The evaluation showed that an edutainment approach, combining short videos and text elements, and a variety of simple digital games, can be promising for improving SE awareness in our target population stratum. Since this target age group is already familiar with gaming and utilizes gaming relatively frequently, the approach appears reasonable for reaching a larger populace.

A challenge remains in making secondary students aware of the availability the edutainment and motivating them to utilize it on a direct or individual basis. While social networks appear feasible for raising awareness, in the face of the abundance of entertainment and game options available, we believe that the most promising approach will likely be informational publicity campaigns towards informatics teachers in secondary schools where groups (i.e., interest groups or classrooms) utilize the software together in a structured setting.

Future work will include public access on the university website, enabling integrated data collection and web analytics to provide further insights into how users became aware of the edutainment, which games were utilized for how long, the number of return visitors, and the inclusion of online surveys. Additionally, the SE pool of questions will be expanded and question and answer placement randomized. A point and badge ranking of top players may provide a separate incentive and motivation for certain player types.

ACKNOWLEDGMENT

The author thanks Carsten Lecon, Christian Wilhelm, Flamur Kastrati, and Lisa Philipp for their assistance with the concepts, realization, and graphics.

REFERENCES

[1] J. Vegso, "Interest in CS as a Major Drops Among Incoming Freshmen," Computing Research News, vol. 17, no.3, 2005.

[2] B. Schmidt, http://benschmidt.org/Degrees/ 2015.07.04

[3] U. Hanselmann, "Die große Kraft," Engl: "The major force", Die Zeit, No. 22, 2009. http://www.zeit.de/2009/22/C-Faecherportraet-Informatik/komplettansicht 2015.07.04,

[4] Bitkom, "In Deutschland fehlen 41.000 IT-Experten," 2014. https://www.bitkom.org/Presse/Presseinformation/Pressemitteilung_1704.html 2015.07.10

[5] D. Parnas, "Software engineering programs are not computer science programs," Software, IEEE, 16(6), pp. 19-30, 1999.

[6] P. Bourque and R. Fairley, "Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0," IEEE Computer Society Press, 2014..

[7] C. Crawford, "The art of computer game design," McGraw-Hill/Osborne Media, 1984.

[8] C. Abt, "Serious Games," The Viking Press, 1970.

[9] T. Connolly, M. Stansfield, and T. Hainey, "An application of games-based learning within software engineering," British Journal of Educational Technology, 38(3), pp. 416-428, 2007.

[10] C. Caulfield, J. Xia, D. Veal, and S. Maj, "A systematic survey of games used for software engineering education," Modern Applied Science, 5(6), 28-43, 2011.

[11] L. Carter, "Why students with an apparent aptitude for computer science don't choose to major in computer science," SIGCSE Bulletin, ACM, vol. 38, no. 1, Mar. 2006, pp. 27-31.

[12] S. Kent. "The Ultimate History of Video Games," Three Rivers Press, 2001.

[13] L. Given (Ed.), "The Sage encyclopedia of qualitative research methods," Sage Publications, 2008.

[14] W. IJsselsteijn, K. Poels, and Y. De Kort, "The Game Experience Questionnaire: Development of a self-report measure to assess player experiences of digital games," TU Eindhoven, Eindhoven, The Netherlands, 2008.

 ISBN: 978-1-61208-438-1

Using Cloud Services To Improve Software Engineering Education for Distributed Application Development

Jorge Edison Lascano[1,2], Stephen W. Clyde[1]
[1]Computer Science Department, Utah State University, Logan, Utah, USA
[2]Departamento de Ciencias de la Computación, Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador
email: edison_lascano@yahoo.com, Stephen.Clyde@usu.edu

***Abstract*—Software-engineering education should help students improve other development skills besides design and coding. These skills, referred to here as A2R (Analysis to Reuse), include analysis, technology evaluation, prototyping, testing, and reuse. The need for improved A2R skills is particularly pronounced in advanced areas like distributed application development. Hands-on programming assignments can be an important means for improving A2R skills, but only if they focus on the right details. This paper presents a case study of programming assignments offered in a graduate-level class on distributed application development, where the assignments required the students to use cloud services and programming tools that were heretofore unfamiliar to the students. Direct observation by the instructor and a post-class survey provided evidence that the assignments did in fact help students improve their A2R skills. The post-class survey also yielded some interesting insights about the potential impact of well-designed programming assignments, which in term led to ideas for future research.**

Keywords-computer science education; software-engineering education; cloud computing; virtual environments; distributed systems.

I. INTRODUCTION

Imagine yourself at a worktable with four or five of your peers. In the center of the table is a pile of seemingly random objects, including two dozen sheets of paper, paper clips, a small roll of tape, pins, and several small wooden sticks. A quick glance around the room reveals a dozen other groups just like yours with similar piles in front of them. An individual, who is introduced as your customer, stands at the front of the room and says that you have 30 minutes to build a "great" tower. What do you do first? How do you put all that you know about paper, clips, tape, wooden sticks, etc. into practice to satisfy the customer's request for a tower and do so within 30 minutes?

Such is the typical scene on the first day of class in the undergraduate introductory course on software engineering at Utah State University (USU). In general, all the students have a good working knowledge of objects at their disposal and even some inkling on how they may combine several of them to create new more structural useful objects. Most groups succeed in creating something that stands on its own and roughly resembles a tower within 30 minutes. However, at the end of that time, the customer surprises the students by giving them a few more objects, e.g., more paper and tape, and asks them to take 15 more minutes to make their towers taller or stronger. Many groups fail to do so in the limited allotted time. In fact, about half of them end up destroying their original towers in the attempt.

Afterwards the instructors and students discuss the experience in terms of what worked well for the group, particular difficulties that hindered progress, how the group organized itself, and how they decided on an overall approach. The discussion usually leads to some very interesting comparisons with common aspects of software engineering, such as group work, tool evaluation, prototyping, design patterns, testing, extensibility, reuse, and more. Over the years, one of the authors, who is a long-time instructor for this introductory software engineering course, has observed the following:

1. Virtually no student or group ever asks the customer what a "great" tower means. Most assume that they already know and proceed to build without each researching the requirements.
2. Virtually no student or group ever looks around to see what other groups have done or are doing, evaluates the ideas they see, and then tries to adapt or improve on them.
3. Only a small percentage of the groups try prototyping an idea to explore its characteristics.
4. Only rarely does a group test the properties (e.g., stability or strength) of a component or the whole tower and then try to make modifications to improve those properties.
5. Only a few groups try to establish patterns or "best practices" either in their building processes or the components they create, and then reuse those ideas.

Each of these observations represents a potential engineering pitfall or negative practice that can lead to inefficiency or failure. Software-engineering education needs to help students avoid these and other related pitfalls by connecting theory with best practices in the context of real non-trivial problems [1]. Doing so goes well beyond teaching the "How To's" of a specific technology, like a programming environment. Instead, it requires educators and students alike to address the "How To's" of the overall development process, including:

1. How do we know when we understand the customer's problem sufficiently?
2. How can we benefit from existing technology or from what others have tried in the past?
3. How can we prototype part of a problem or alternative solutions to answer critical questions?

 ISBN: 978-1-61208-438-1

4. How can we test what we build?
5. How can we find good solutions to reoccurring problems and reuse that knowledge?

More concisely, software-engineering education needs to help students make analysis, technology evaluation, prototyping, testing, and reuse an effective and integral part of their development activities [1]. Here, we'll refer to these as Analysis to Reuse (A2R) skills.

The need for better A2R skills is prevalent in every software-engineering domain, but is pronounced in the development of distributed applications. Distributed-application development, or distributed-system development at large, has all of the challenges of traditional software development, plus the complexities introduced by inter-process communications, concurrency, the potential of partial failure, and replication that exist for performance improvements or fault tolerance [2].

Now let us roll our classroom scene forward several years to a graduate software-engineering class that focuses on distributed applications. Students entering in this class have solid foundations in software-engineering fundamentals, programming languages, inter-process communications using sockets, and many other areas of computer science. Yet, they still need to strengthen their A2R skills, especially in the context of distributed applications, and the best way to do that is through hands-on experience [3]. So, from an education perspective, the challenge is to provide realistic and engaging assignments that will strengthen the A2R skills and are doable within the allotted time.

Because distributed applications are relatively complex [4] by their very nature, there are two negative tendencies for program assignments in this area: a) abstracting away too many interesting aspects of the problem and b) getting bogged down with unnecessary application-domain details.

The first tendency is very common in advanced CS courses, because simpler assignments are more manageable, teachable, and easier to fit within a given allotted time. Advanced courses usually have to operate within same time constraints as introductory courses. Even though, they are more complex, it is essential that advanced assignments include reasonable limits on the expected time and effort [4]. Simplicity in their design is a necessity and by itself is not a problem. Focusing on the wrong details and abstracting away all interesting parts of the problem, however, is a serious real pitfall. For example, scalability is a real and very common aspect of most distributed applications [2]. Even though removing scalability requirements would simplify an assignment, it would rob the students of a valuable opportunity to improve A2R skills in a relevant area.

The second tendency is to allow an assignment to get bogged down in application-domain details, shifting focus away from the learning objectives. Assignments in advanced courses, like distributed-application development, work best if they are grounded in a meaningful real-world domain. However, most distributed applications and their domains are relatively complex. If not careful, an instructor could easily use all available time explaining the sample application domain, instead about the core course topics. Keeping assignments focused on a small set of functional requirements that require minimal application-domain knowledge, is essential to making sure that they are doable within time limits and achieve the learning objectives.

This paper describes a case study of programming assignments conducted in an advanced software-engineering class on distributed-application development, where all of the assignments required students to use cloud resources for their execution environment. The hoped-for result was that the assignments provided students significant opportunities to improve their A2R skills, while introducing them to new concepts and development tools. Section 2 describes the course's programming assignments in terms of their learning objectives, the application domains that act as backdrops, and their requirements. Section 2 also explains the tools and technologies introduced for each assignment. Section 3 summarizes the instructor's observations made throughout the semester and assignment design learnings. To evaluate the effectiveness of the assignments, we conducted a post-class survey. Section 4 describes this survey and presents the resulting raw data. Since the class was a second-year graduate class, the enrollment was small. So, we cannot make many generalizations from the survey data. Nevertheless, they do lead us to some interesting insights. We share those insights in Section 4.B. Section 5 explores related work in software-engineering education using cloud resources and hands-on learning. Finally, Section 6 provides conclusions, along with ideas for future research that could further advance software-engineering education relative to A2R skill development.

II. Programming Assignment For a Distributed Application Development Course

CS 6200 at USU is a second-year graduate course in software engineering that focuses on the development of distributed applications. Its prerequisite, CS 5200, provides students with a strong foundation in inter-process communication, protocols, concurrency, and communication subsystems. CS 5200 is also a programming intensive course, which means that students who successfully complete it have confidence in their ability to implement non-trivial software systems. The overall learning objectives for CS 6200 are as follows:

- Master theoretical elements of distributed computing, including: models of computation and state, logical time, vector timestamps, concurrency controls, and deadlock;
- Become familiar with the provisioning and use of virtual computational and storage resources in a cloud environment;
- Become familiar with cloud-based tools for processing large amounts of data efficiently; and
- Become familiar with distributed transactions and resource replication.

For the Spring 2015 semester, the homework was broken down into five assignments, each lasting two to three weeks.

 ISBN: 978-1-61208-438-1

A. Assignments 1 & 2 – Disease Tracking System

For the first two assignments, the student implemented a set of processes that worked together to form a disease tracking and outbreak monitoring system. They had to deploy multiple processes on EC2 instances within Amazon Web Services (AWS) cloud. The first type of processes were simulations of Electronic Medical Records Systems (EMR's) that randomly generated notifications of diagnoses for infectious diseases, like influenza. The EMR's sent these disease notifications to Health District Systems (HDS's), which collated diagnoses and then sent periodic disease counts to Disease Outbreak Analyzers (DOA's). Each DOA monitored outbreaks for a single type of disease. See Figure 1. The specific learning objectives for these two assignments included:

- Review inter-process communications;
- Become familiar with vector timestamps and how they behave in a distributed system under varied conditions;
- Become familiar with setting up and using computational resources in a cloud, e.g., AWS; and
- Become familiar with setting up a simple name service.

The students were asked to learn and use Node.js as the primary programming framework [5][6]. Because Node.js was new to all the students, some class time was dedicated to teaching Node.js, but only enough to get them started. Their unfamiliarity with Node.js was also the reason this first system was split into two assignments. They built and tested approximately half of the functionality in the first assignment and the remainder in the second.

To deploy their systems to EC2 instances on AWS, the students had to learn about security on AWS, create security keys, and setup their own user accounts using Amazon's Identity and Access Management (IAM). They also had to setup and learn the AWS's command-line language interface (AWSCLI), so they could automate the deployment and launching of their systems.

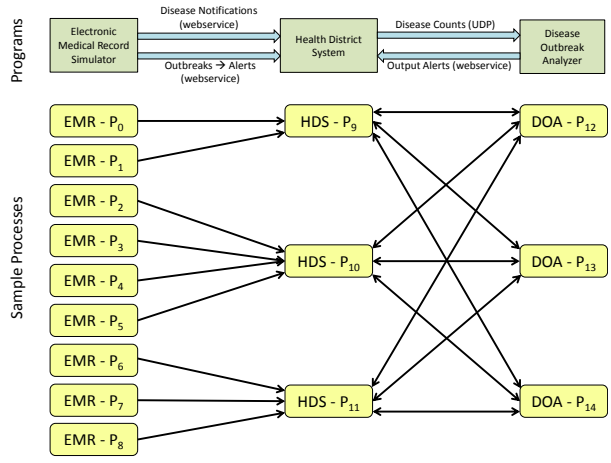


Figure 1. Programs built as part of Assignments 1 & 2, plus an illustration of sample processes.

B. Assignment 3 – Twitter Feed Analysis

In this assignment, the students explored how to process big data using MapReduce on AWS and how to configure cloud resources using AWS's Cloud Formation tools. Specifically, they were to capture tweets through Twitter's API and then analyze them for positive or negative sentiment relative to some key phrase, like "health care". The learning objectives for this assignment were as follows:

- Become familiar with setting up and using MapReduce with a cloud-based distributed file system;
- Become familiar with tools for provisioning collections of resources that are needed for a distributed system; and
- Explore the types of problems that are well suited for a MapReduce solution

To complete this assignment, students setup and learned how to use AWS's S3, MapReduce, and Cloud Formation services. Some also used this assignment to learn about a Node.js module for working directly with AWS; while others strengthened their knowledge of AWSCLI.

C. Assignment 4 – Distributed Election

In this assignment, the students implemented a distributed system consisting of dozens of processes that shared access to common data files, which were collectively treated as one large shared resource, like a database. One of the processes played the role of Resource Manager (RM) and accessed the common data files in response to requests from the other processes. If RM died, then the other processes had to detect that failure and elect one of the remaining processes to be the new RM seamlessly. The learning objectives for this assignment were:

- Master at least one distributed election algorithm;
- Master the concept of resource managers for controlling access to share resources; and
- Become more familiar with tools for provisioning collections of resources in a cloud.

To complete this assignment, we allowed students to use any of the tools they had learned thus far, but they had to deploy their systems to multiple EC2 instances and demonstrate that the system would elect a new RM if the current one was stopped. They had to show that the system has as a whole, lost no work.

D. Assignment 5 – Distributed Transactions

In this assignment, the students had to build a simple transaction management system with locking capabilities. Like Assignment 4, this system had to support multiple concurrent worker processes, but went a step further in requiring multiple shared resources and multiple concurrent RM's. Each RM had to keep track of a single resource and support lock, read, write, and unlock operations on that resource. The system also had to include a transaction manager that supported starting, committing, and aborting of transactions. Assignment 5's learning objectives included:

- Become familiar with locking; and

 ISBN: 978-1-61208-438-1

- Become familiar with transaction management in a distributed system.

Like Assignment 4, the students could use any of the tools that they learned to this point in completing Assignment 5.

III. INSTRUCTOR OBSERVATIONS

Seven students took CS 6200 in the Spring 2015 semester: 5 who were registered for credit and 2 who audited the class. It's impossible to recap all that took place during the semester, but we summarize a few observations prior to presenting the post-class survey to help set the stage for the survey and our conclusions.

First, we observed that all of seven students started the class with roughly equivalent software-engineering backgrounds and programming skills, even through they were not all seeking the same degree nor did they have the same programs of study. None of the students had used Node.js before and only one had any exposure to cloud computing, and that was only a light exposure.

Second, we observed that requiring students to setup and managing their own cloud resources not only helped them with core concepts and development skills, but it also allowed them to improve their A2R skills relative to figuring out what the most important requirements were, tool evaluation, and testing. For example, in the first two assignments, the students had to deploy their system to EC2 instances. For most of the students, this was the first time deploying something that they built to an execution environment different from their development environment, along an execution environment consisting of multiple virtual machines. It opened their eyes to new challenges, such as firewall issues, file permissions, and missing dependencies. Time was made available in every class period for them to talk about the challenges that they were facing and get ideas from other students or the instructor about how to address those challenges. Similar discussion also took place on an online forum. By the end of Assignment 2, the classroom and online discussions showed that the students had stepped up their efforts to understand the assignment requirements, evaluate the tools available to them, and test their work.

Even though the purposes of Assignments 4 and 5 were considerably different from the first three, they possessed some of the same challenges, like resource name resolution and deployment into a cloud environment. It was encouraging to see that the students solved these problems by adapting techniques used in the earlier assignments and improving upon them – evidence of them practicing A2R skills.

We were happy to see that the students learned some unexpected, but very relevant lessons. For example, one student stored his access keys in a text file and committed that file to a public Git repository. It wasn't long before someone hacked his AWS account. Amazon and the student caught the problem relatively quickly and simultaneously, but not before the hacker had used over $600 of resources. He ended up taking extra time learning more about security from unauthorized use. Thankful, Amazon worked with him to recover the expenses, so he did not have to pay for the lost out of pocket. Still, it proved to be a valuable learning experience that he will not forget.

With respect to the selected cloud AWS, we observed that it provided a mature and full-featured set of services for the students to learn from. In some areas, AWS's learning curve was steeper than necessary, but with supplementary examples and good discussions, it was manageable. From an education perspective, a good thing about AWS is that it has features in three main categories: Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS) [2].

One negative experience with AWS occurred during Assignment 3, which depended on an Amazon-provided template for setting up a MapReduce cluster. That template was changed by its authors in the middle of assignment, causing several of the students not to complete all of the requirements. To avoid this problem in the future, the instructors will make private copies of public or external resources, so changes to them will not affect assignments in progress.

A. Assignment Design Learning

When instructors design assignments oriented to networking or distributed applications, they need to consider distribution concepts, but at the same time bear in mind the limitations for the students' capabilities and hardware environment. Before cloud resources became available, this typically consisted of one computer [7] or small number of computers on a local area network (LAN) in a school lab. Assignments that work well on one computer or a LAN may not allow the students to gain appreciation for more realistic networking challenges, performance issues, and reliability problems [8]. With cloud resources, assignments can now be designed having a broad range of resources in mind, while still considering good software-engineering practices for analysis, technology evaluation, testing, deployment and even reuse.

IV. POST-CLASS SURVEY

To assess the value of the programming assignments for CS 6200, we designed a post-class survey and conducted that survey with two populations: students registered in CS 6200 for credit and students who just audited the class. Those registered for credit had to complete all of the assignments to receive a grade; those just auditing the class did not. In fact, it is important to note that none of the second group completed any assignment.

A. Survey design

We organized the survey into two parts. The first part asked students to rate their knowledge and skills in areas related to the course and the assignments, as they were before the class started, using a 1-to-5 scale. The second part asked them to do the same relative to the end of class. *Table*

 ISBN: 978-1-61208-438-1

and *Table* list the concepts (knowledge areas) and skills respectively covered in both parts of the survey. The survey used a *Matrix Table* format, with the concepts and skills as rows and possible ratings as columns. See Figure 2 for partial view of the survey instrument for Part 1.

The difference between each individual's answers to corresponding questions from two parts provides a glimpse of that person's perceived change in knowledge or skill levels as a consequence of the course.

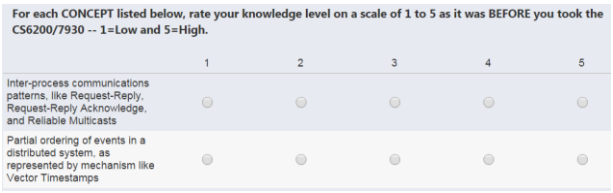


Figure 2. Partial view of the survey.

We could have administered a pre-class survey similar to the first part, but considerable differences in each student's personal rating scheme would likely have evolved over the semester, making it difficult to assess perceived change. We could have also administered pre and post exams to measure their proficiency objectively, but there was no common knowledge basis for a pre exam. So, the study would have degenerated into the interpretation of just post exam results.

TABLE I. KNOWLEDGE SURVEY QUESTIONS.

No	Knowledge/Concept	Acronym
Q1.1	Inter-process communications patterns, like Request-Reply, Request-Reply Acknowledge, and Reliable Multicasts	IPC
Q1.2	Partial ordering of events in a distributed system, as represented by mechanism like Vector Timestamps	VTS
Q1.3	Message serialization/deserialization	S/D
Q1.4	Intra-process concurrency	IntraPC
Q1.5	Computation resources in a cloud-computing environment, such as AWS	AWS
Q1.6	Namespaces, name services, and name resolution	NS
Q1.7	Deployment, execution and testing techniques in a distributed environment	Deploy
Q1.8	Deployment, execution and testing techniques in the cloud.	Testing
Q1.9	Distributed election algorithms	DEA
Q1.10	Resource managers	RM
Q1.11	Fault tolerance in a distributed environment.	FT
Q1.12	Tools for provisioning collection of resources needed for a distributed system.	Tools
Q1.13	Cloud Computing resources	CCR
Q1.14	Infrastructure as a Service (IaaS)	IaaS
Q1.15	Platform as a Service (PaaS)	PaaS
Q1.16	Inter-process concurrency	InterPC

TABLE II. SKILLS SURVEY QUESTIONS.

No	Skill	Acronym
Q2.1	AWS Users and key pairs (Identity and Access management -IAM)	AWS-IAM
Q2.2	AWS Virtual PC Instances (EC2)	AWS-EC2
Q2.3	AWS Storage (S3, EBS)	AWS-S3,EBS
Q2.4	AWS-CLI (Command Line Interface)	AWS-CLI
Q2.5	AWS SDK (Software Development Kit)	AWS-SDK
Q2.6	Managing instances in AWS: creating/launching, starting, stopping, terminating	EC2-Instances
Q2.7	AWS Billing	AWS-Billing
Q2.8	Using Node.js to Develop Distributed Systems	Node.js_DS
Q2.9	Using Node.js to deploy and run Distributed Systems in the cloud	Node.js_Cloud
Q2.10	Designing and developing TCP/UDP/Web Services-based systems with Node.js	Node.js_C/S
Q2.11	Writing scripts to Deploy/execute applications in distributed environments	DS_Scripts
Q2.12	Designing and Developing Resource Managers	RM_DD
Q2.13	Designing and Developing Distributed Election Algorithms	DEA_DD

B. Survey Results

All seven students completed both parts of the survey. Figures 3 and 4 show averages of the students' raw estimates of their knowledge and skill levels for before and after class. The blue lines represent the levels before and the red lines after. The (a) graphs are for the first population, namely the students who registered for credit and the (b) graphs are for the auditing students. Figure 5 shows the average net change in the levels, broken down by the two populations.

One interesting result that is worth pointing out immediately, is that the first group of students, in general, rated their before-classes level lower than the second population. We believe that this can be contributed to the common adage, "You don't know what you don't know". The first group of students did the assignments and soon discovered how much they really didn't know, whereas the second group did not come to the same realization. For example, the auditors' perception about their AWS and Node.js skills was that they knew those technologies relatively well before starting the class; meanwhile the first group of students came to realize that their skills were almost nil.

Next, notice that the estimated pre-class knowledge levels are higher than the estimated pre-class skill levels. In general, the students felt they had a conceptual understanding of the course concepts, including AWS, which only one student had exposure to before class. From this, we can see that students (and perhaps all people) tend to believe that they are able to generalize conceptual knowledge into new areas that they have not seen before.

 ISBN: 978-1-61208-438-1

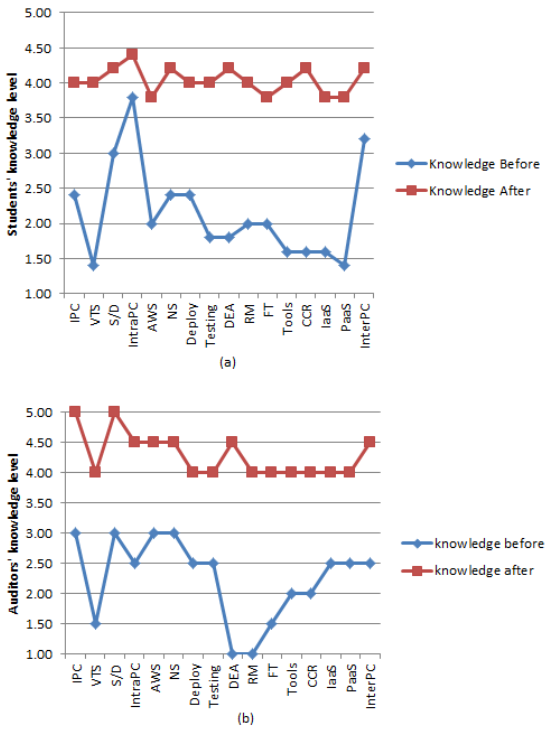


Figure 3. Before and After Knowledge Levels.

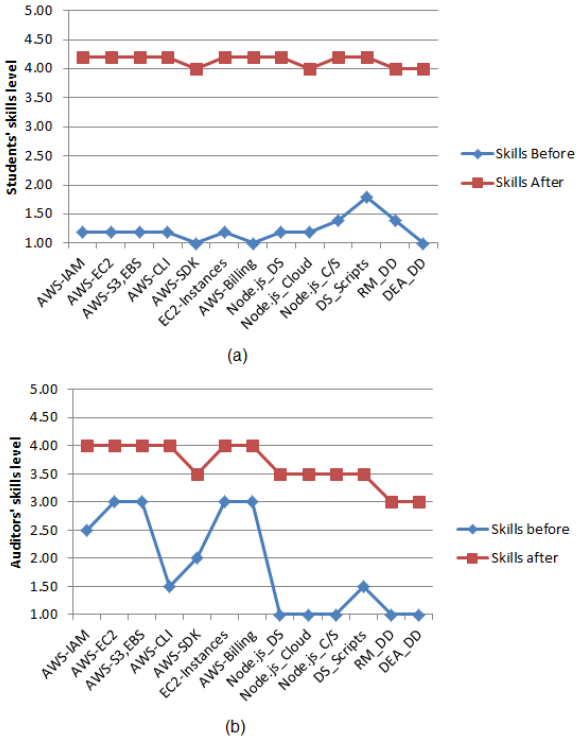


Figure 4. Before and After Skills Levels.

Figure 5 shows evidence that the first group of students truly improved their skills. Their net change for every skill was higher than the net change for the second group. Interestingly, the same is not true in the knowledge area. At first glance, this might seem odd, but considering the timing and relative nature of the self-made estimates, there is a possible explanation. Specifically, the students who didn't do the assignments naturally felt that their biggest growth was in increase of conceptual knowledge.

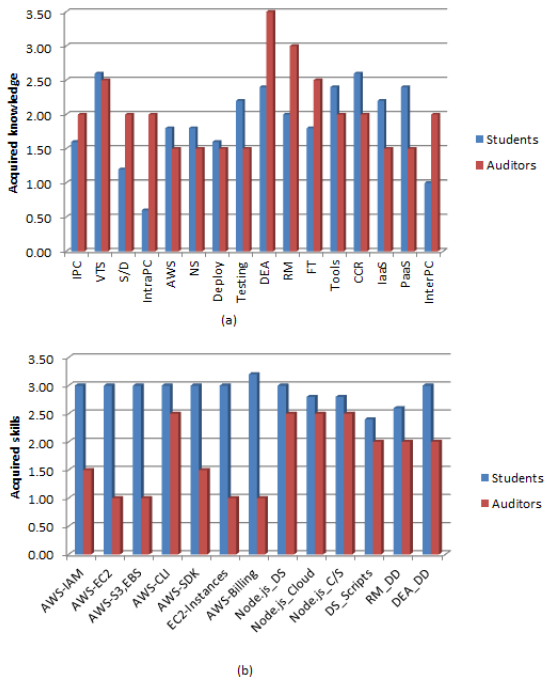


Figure 5. Perception of acquired knowledge: differences between the after and the before.

V. RELATED WORK

Other higher-education institutions are using cloud computing resources in courses that focus on distributed applications or network programming. Clearly, these platforms allow the students to use realistic testing and production environments. Moreover, there are large research universities that have implemented private clouds on their campuses and use them in the classroom. For example, Syracuse University provides a local virtual machine lab used to form virtual networks for security projects [9]. North Carolina State University supplies computing resources over the Internet with their Virtual Computing Lab [10], Arizona State University developed V-lab for Networking Courses [3], and Okanagan College and King's university College talk about using a cloud for educational collaboration [11]. Nevertheless, these private solutions are often not economically viable for many universities [7], and therefore they can only consider public cloud solutions.

Programming assignments that use public or private clouds can add value to the learning experience and increase students' skills directly related to possible professional careers [4] in network programming [7], distributed systems [11], systems administration [4], security [4][9], data processing [12], among others. Furthermore, a major benefit is that students do not need to simulate network

communications over a localhost interface [7]; instead, they can use multiple virtual machines and real network communications to better understanding the distributed system components, their roles, and the related concepts.

Using a public cloud for hands-on activities offers benefits such as scalability, flexibility, security, cost-efficiency and accessibility [7], which all are key characteristics of distributed systems [2]. Public clouds also add an interesting and valuable dimension to the execution and debugging of distributed applications [12], without needing huge budgets for private-cloud or physical-machines infrastructure. Most of the public cloud providers, e.g., Amazon, Google, Microsoft, IBM, offer grants for academic institutions that want to use their resources for educational purposes. For example, at the time of this study, Amazon offered grants up to $100 per students [13]. Other benefits to public clouds include ready access to different operating-system platforms, communication protocols, development tools, open-source code, public forums, and more.

VI. CONCLUSIONS AND FUTURE WORK

For this small case study, we conclude that programming assignments with requirements to use cloud resources were successful in helping the CS 6200 students to improve their A2R skills, as well as their core distributed-application development skills. Both the instructor's observations and the post-class survey provide anecdotal evidence of their improvement.

We also found some evidence that students are willing and even excited to learn new tools and skills, especially if they can see how it lets them put theory into practice. Even though the assignments were based on carefully crafted and sanitized requirements, they were realistic enough for the students to experience real problems and see how theoretical concepts, like vector timestamps and distributed election, could be used to solve those real problems.

Some important design criteria for assignments included: a) hiding unnecessary details, like all the other capabilities of an EMR beside the generation of disease notifications, b) focusing on requirements that put theory into practice, like the election of an RM in Assignment 4, c) including non-trivial non-functional requirements, like scalability, and d) wherever possible allow students to reuse components or knowledge acquired in previous assignments.

The survey data also opened some doors to possible future research. Specifically, we would like to conduct a broader experiment across multiple software-engineering classes of various kinds and at different levels, to explore specific ways that the design of assignments can improve A2R skills in general. From that, we hope to publish more concrete guidelines for programming-assignment design for software-engineering classes at all levels.

ACKNOWLEDGMENT

We would like to thank Amazon for providing funding for the students to use AWS resources for this class.

REFERENCES

[1] C. Ramamoorthy, "Computer Science and Engineering Education," IEEE Transactions on Computers, Vols. C-25, no. NO. 12, December 1976, pp. 1200-1206.

[2] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, Distributed Systems: Concepts and Design, 5th ed., Boston, MA: Addison-Wesley Publishing Company, 2011, p. 1008.

[3] L. Xu, D. Huang, and W.-T. Tsai, "A Cloudbased Virtual Laboratory Platform for Hands-On Networking Courses," in ITiCSE '12 the 17th ACM annual conference on Innovation and technology in computer science education, New York, 2012, pp. 256 - 261.

[4] C. Leopold, Parallel and distributed Computing, New York: John wiley & Sons, Inc., 2001.

[5] C. Gonzalez, C. Border, and T. Oh, "Teaching in Amazon EC2," in SIGITE'13, Special Interest Group for Information Technology Education, Orlando, Florida, 2013, pp. 149-150.

[6] D. Howard, Node.js for PHP Developers, S. S. L. a. M. Blanchette, Ed., Sebastopol, CA: O'Reilly Media, Inc, 2012.

[7] node.js, "node.js," Joyent, 2015. [Online]. Available: https://nodejs.org/. [Accessed 28 04 2015].

[8] W. Zhu, "Hands-On Network Programming Projects in the Cloud," in SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education, New York, 2015, pp. 326-331.

[9] j. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: A Platform for Educational Cloud Computing," in 40th ACM technical symposium on Computer science education SIGCSE'09, New York, 2009, pp. 111-115.

[10] W. Du and R. Wang, "SEED: A Suite of Instructional Laboratories for Computer Security Education," Journal on Educational Resources in Computing (JERIC), vol. 8, no. 1, March 2008, pp. 3:1-3:24.

[11] H. E. Schaffer, S. F. Averitt, M. I. Hoit, A. Peeler, E. D. Sills, and M. A. Vouk, "NCSU's Virtual Computing Lab: A Cloud Computing Solution," Computer, vol. 42, no. 7, July 2009, pp. 94 - 97.

[12] Y. Khmelevsky and V. Voytenko, "Cloud computing infrastructure prototype for university education and research," in WCCCE '10, 15th Western Canadian Conference on Computing Education, New York, 2010.

[13] A. Rabkin, C. Reiss, R. Katz, and D. Patterson, "Using clouds for MapReduce measurement assignments," ACM Transactions on Computing Education (TOCE), vol. 13, no. 1, January 2013, pp. 2:1-2:18.

[14] AWS, "AWS in Education Grants," Amazon, [Online]. Available: http://aws.amazon.com/grants/. [Accessed 01 07 2015].

 ISBN: 978-1-61208-438-1

Controlled Variability Management for Business Process Model Constraints

Neel Mani

ADAPT Centre for Digital Content Technology
Dublin City University, School of Computing
Dublin, Ireland
Email: nmani@computing.dcu.ie

Claus Pahl

ADAPT Centre for Digital Content Technology
Dublin City University, School of Computing
Dublin, Ireland
Email: cpahl@computing.dcu.ie

***Abstract*—Business process models are abstract descriptions that are applicable in different situations. To allow a single process model to be reused, configuration and customisation features can help. Variability models, known from product line modelling and manufacturing, can control this customisation. While activities and objects have already been subject of similar investigations, we focus on the constraints that govern a process execution. We report here on the development a rule-based constraints language for a workflow and process model. The aim is a conceptual definition of a domain-specific rule variability language, integrated with the principles of a common business workflow or process notation. This modelling framework will be presented as a development approach for customised rules through a feature model. Our use case is content processing, represented by an abstract ontology-based domain model in the framework.**

Keywords–Business Process Modelling, Process Constraints, Variability Model, Domain-specific Rule Language.

I. INTRODUCTION

Business process models are abstract descriptions that can be applied in different situations and environments. To allow a single process model to be reused, configuration and customisation features help. Variability models, known from product line engineering, can control this customisation. While activities and objects have already been subject of customisation research, we focus on the customisation of constraints that govern a process execution here. Specifically, the recent emergence of business processes as a services in the cloud (BPaaS) highlights the need to implement a reusable process resource together with a mechanism to adapt this to consumers.

We are primarily concerned with the utilisation of a conceptual domain model for business process management, specifically to define a domain-specific rule language for process constraints management. We present a conceptual approach in order to define a Domain Specification Rule Language (DSRL) for process constraints [1] based on a Variability Model (VM). To address the problem, we follow a feature-based approach to develop a domain-specific rule language, borrowed from product line engineering. It is beneficial to capture domain knowledge and define a solution for possibly too generic models through using a domain-specific language (DSL). A systematic DSL development approach provides the domain expert or analyst with a problem domain at a higher level of abstraction. DSLs are a favourable solution to directly represent, analyse, develop and implement domain concepts. DSLs are visual or textual languages targeted to specific problem domains, rather than general-purpose languages that aim at general software problems. With these languages or models, some behaviour inconsistencies of semantics properties can be checked by formal detection methods and tools.

Our contribution is a model development approach using a feature model to bridge between a domain model and the domain-specific rule extension of a business process to define and implement process constraints. The feature model streamlines the constraints customisation of business processes for specific applications. The novelty lies in the use of software product line technology to customise processes.

We choose content processing here as a specific domain context to illustrate the application the proposed domain-specific technique (but also look at the transferability to other domains in the evaluation). We use a text-based content process involving text extraction, translation and post-editing as a sample business process. Note that we have also investigated the subject in the context of e-learning processes, which we will also address later on. We also briefly discuss a prototype implementation. However, note that a full integration of all model aspects is not aimed at as the focus here is on models. The objective is to outline principles of a systematic approach towards a domain-specific rule language for content processes.

The paper is organised as follows. We review process modelling and constraints in Section 2. In Section 3, we content processing from a feature-oriented DSL perspective. Section 4 introduces rule language background and ideas for a domain-based rule language. We then discuss formal process models into with the rule languages can be integrated.

II. BUSINESS PROCESS MODELS AND CONSTRAINTS

At the core is a process model that defines possible behaviour. This is made up of some frame of reference for the system and the corresponding to the attributes used to describe the possible behaviour of the process. The set of behaviours constitutes a process referred to as the extension of the process and individual behaviours in the extension are referred as instances. Constraints can be applied at states of the process to determine its continuing behaviour depending on the current situation. The rules combine a condition (constraint) on a resulting action. The target of our rule language (DSRL) is

 ISBN: 978-1-61208-438-1

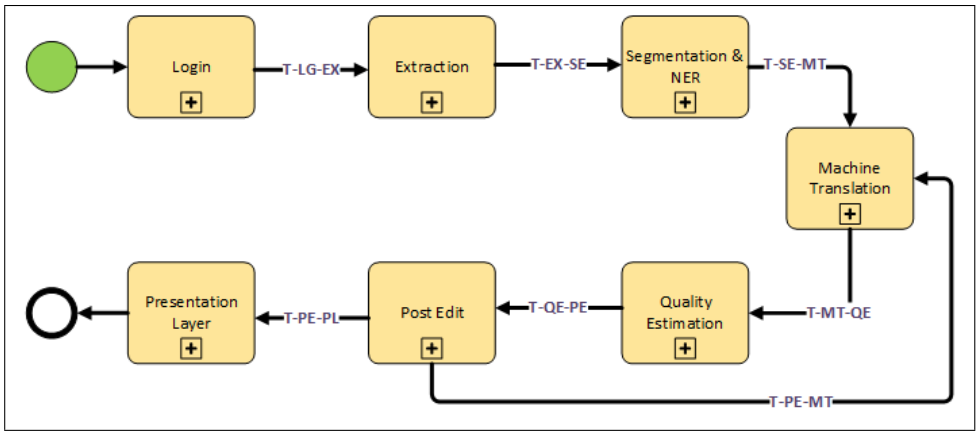


Figure 1. Workflow design of content process.

a standard business process notation (as in Fig. 1). Rules shall be applied at the processing states of the process.

Our application case study is intelligent content processing. Intelligent content is digital content that allows to users to create, curate and consume content in a way that satisfies dynamic and individual requirements relating to task design, context, language, and information discovery. The content is stored, exchanged and processed by a Web architecture and data will be exchanged, annotated with meta-data via web resources. Content is delivered from creators to consumers. Content follows a particular path which contains different stages such as extraction and segmentation, name entity recognition, machine translation, quality estimation and post-editing. Each stage in the process has its own challenges and complexities.

We assume the content processing workflow as in Figure 1 as a sample process for the rule-based instrumentation of processes. Constraints govern this process. For instance, the quality of a machine-based text translation decides whether further post-editing is required. Generally, these constraints are domain-specific, e.g., referring to domain objects, their properties and respective activities on them.

III. Domain and Feature Model

Conceptual models (CM) are part of the analysis phase of system development helping to understand and communicate particular domains [1]. They help to capture the requirements of the problem domain and, in ontology engineering, a CM is the basis for a formalized ontology. We utilise a conceptual domain model (in ontology form) to derive a domain-specific process rule language. A domain specific language (DSL) is a programming or specification language that supports a particular application domain through appropriate notation, grammar and abstractions [2]. DSL development requires both domain knowledge and language development expertise. A prerequisite for designing DSLs is an analysis that provides structural knowledge of the application domain.

A. Feature Model

The most important result of a domain analysis is a feature model. A feature model covers both the aspects of software family members, like commonalities and variabilities, and also reflects dependencies between variable features. A feature diagram is a graphical representation of dependences between a variable feature and its components. Mandatory features are present in a concept instance if their parent is present. Optional features may be present. Alternative features are a set of features from which one is present. Groups of features are a set of features from which a subset is present if their parent is present. Mutex and Requires are relationships that can only exist between features. Requires means that when we select a feature, the required featured must be selected too. Mutex means that once we choose a feature the other feature must be excluded (mutual exclusion).

A domain-specific feature model can cover languages, transformation, tooling, and process aspects of DSLs. For feature model specification, we propose the FODA (Feature Oriented Domain Analysis) [3] method. It represents all the configurations (called instances) of a system, focusing on the features that may differ in each of the configurations [4]. We apply this concept to constraints customisation for processes. The Feature Description Language (FDL) [5] is a language to define features of a particular domain. It supports an automated normalization of feature descriptions, expansion to disjunctive normal form, variability computation and constraint satisfaction. It shall be applied to the content processing use case here. The basis here is a domain ontology called GLOBIC (global intelligent content), which has been developed as part of our research centre. GLOBIC elements are prefixed by gic.

Feature diagrams are a FODA graphical notation. They can be used for structuring the features of processes in specific domains. Figure 2 shows a feature diagram for the GLOBIC content extraction path, i.e., extraction as an activity that operates on content in specified formats. This is the first step in a systematic development of a domain-specific rule language (DSRL) for GLOBIC content processing use case. The basic component gic:Content consists of a gic:Extraction element, a mandatory feature. A file is a mandatory component of gic:Extraction and it may either be used for Document or Multimedia elements or both. The closed triangle joining the lines for document and multimedia indicates a non-exclusive (more-of) choice between the elements. The gic:Text has two mandatory states Source and Target. Source contains ExtractedText and Target can be TranslationText. Furthermore, expanding the feature Sentence is also a mandatory component of ExtractedText. The four features Corpora, Phrase, Word

 ISBN: 978-1-61208-438-1

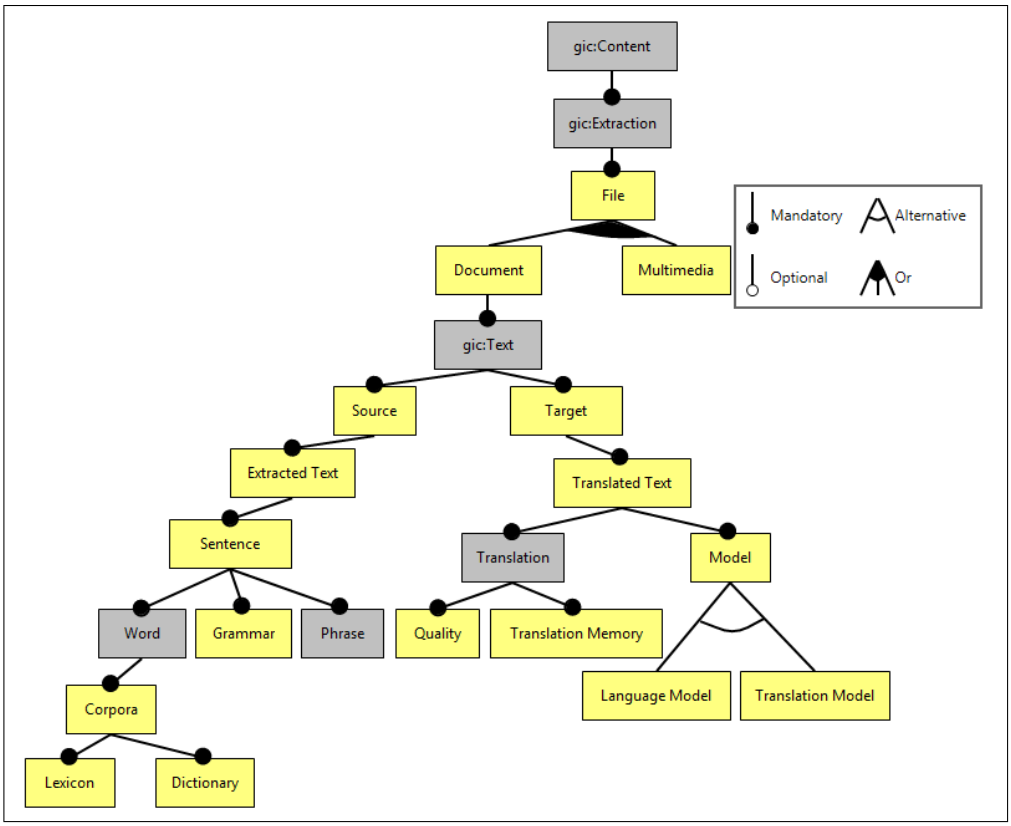


Figure 2. Workflow design of content process.

and Grammar are mandatory. On the other side of gic:Text, a TranslationText is a mandatory component of Target, also containing a mandatory component Translation. A Translation has three components: TranslationMemory and Model are mandatory features, Quality is an optional feature. A Model may be used as a TranslationModel or a LanguageModel or both models at same time. An instance of a feature model consists of an actual choice of atomic features matching the requirements imposed by the model. An instance corresponds to a text configuration of a gic:Text super class. The number of possible gic:Text feature combinations is 512 for the given model, structured and made accessible through the model.

The feature model might include for instance duplicate elements, inconsistencies or other anomalies. We can address this situation by applying consistency rules on feature diagrams. Each anomaly may indicate a different type of problem. The feature diagram algebra consists of four set of rules [4]:

- Normalization Rules - rules to simplify the feature expression by redundant feature elimination and normalize grammatical and syntactical anomalies.
- Expansion Rules - a normalized feature expression can be converted into a disjunctive normal form.
- Satisfaction Rules - the outermost operator of a disjunctive normal form is one-of. Its arguments are All expressions with atomic features as arguments, resulting in a list of all possible configurations.
- Variability Rules - feature diagrams describe system variability, which can be quantified (e.g. number of possible configurations).

The feature model is the key element. Thus, checking internal coherence and providing a normalised format is important for its accessibility for non-technical domain experts. In our setting, the domain model provides the semantic definition for the feature-driven variability modelling.

B. *Domain Model*

Semantic models have been widely used in process management [6,7]. This ranges from normal class models to capture structural properties of a domain to full ontologies to represent and reason about knowledge regarding the application domain or also the technical process domain [8,9]. Domain-specific class diagrams are the next step from a feature model towards a DSL definition. A class is defined as a descriptor of a set of objects with common properties in terms of structure, behaviour, and relationships. A class diagram is based on a feature diagram model and helps to stabilise relationship and behaviour definitions. Note that there is an underlying domain ontology here, but we use the class aspects (subsumption hierarchy only).

In the content use case, class diagrams of gic:Content and its components based on common properties are shown in Figure 3. The class diagram focuses on gic:Text. The two major classes are Text (Document) and Movie files (Multimedia), consisting of different type of attributes like content : string, format : string, or frame rate : int. Figure 3 is the presentation of an extended part of the gic:Content model. For instance, gic:Text is classified into the two subclasses Source and Target. One file can map multiple translated texts or none. gic:Text is multi-language content (source and target content).

 ISBN: 978-1-61208-438-1

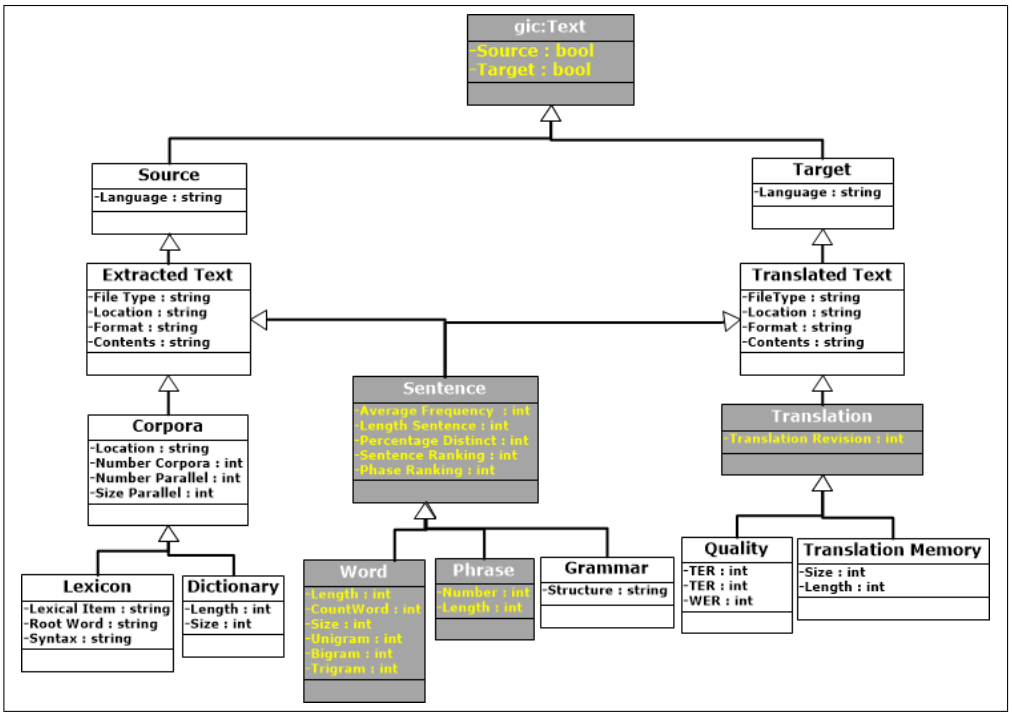


Figure 3. Domain model for global content.

IV. Constraints Rule Language

Rule languages typically borrow their semantics from logic programming [12]. A rule is defined in the form of If-then clauses containing logical functions and operations. A rule language can enhance ontology languages, e.g., by allowing one to describe relations that cannot be described using for instance description logic (DL) underlying the definition of OWL. We adopt Event-Condition-action (ECA) rules to express rules on content processing activities. The rules take the constituent elements of the GLOBIC model into account:

- Content objects (e.g., text) that are processed.
- Content processing activities (e.g., extraction or translation) that process content objects.

An example shall illustrate ECA rules for extraction as the activity. Different case can be defined using feature models:

- We can customise rules for specific content types (text files or multimedia content).
- We can also vary according to processing activities (extraction or translation).

Three sample rule definitions are:

- On uploading file notification from user and if filetype is valid, then progress to Extraction
- On a specific key event and Text is inputted by user and if text is valid then progress Process to Segmentation
- On a specific key event and Web URL input by user and if URL is valid then progress to Extraction and Segmentation

We define the rule language as follows using GLOBIC concepts (examples here):

gicRule ::= [gic:Event] || [gic:Cond] || [gic:Action]
gic:Event::= {Upload} || {Translate} || {Extract}

While the rule syntax is simple, the important aspect is that that the syntactic elements refer to the domain model, giving it semantics and indicating variability points. Variability points are, as explained, defined in the feature model. The above three examples can be formalised using this notation. Important here is thus the guidance in defining rules that a domain expert gets through the domain model as a general reference framework and the feature model definition the variability points.

V. Implementation

While this paper focuses on the conceptual aspects, a prototype has been implemented. Our implementation (cf. Fig. 4) provides a platform that enables building configurable processes for content management problems and constraints running in the Activiti workflow engine. In this architecture, a cloud service layer perform data processing and avail of different resources using the Content Service Bus (based on the Alfresco content management system) to perform activities.

Every activity has its own constraints. The flow of entire activities is performed in a sequential manner so that each activitys output becomes input to the next. The input data is processed through the content service bus (Alfresco) and the rule policy is applied to deal with constraints. The processed data is validated by the validation & verification service layer. After validation, processing progresses to the next stage of the Activiti process - e.g., if Segmentation & Extraction data is validated, then it will automatically move to the Name Entity Recognition stage, otherwise sent for reprocessing.

Reasons to architecturally separate a Service Layer include:

- To provide the capability of grouping, interlinking and coupling services within components.

 ISBN: 978-1-61208-438-1

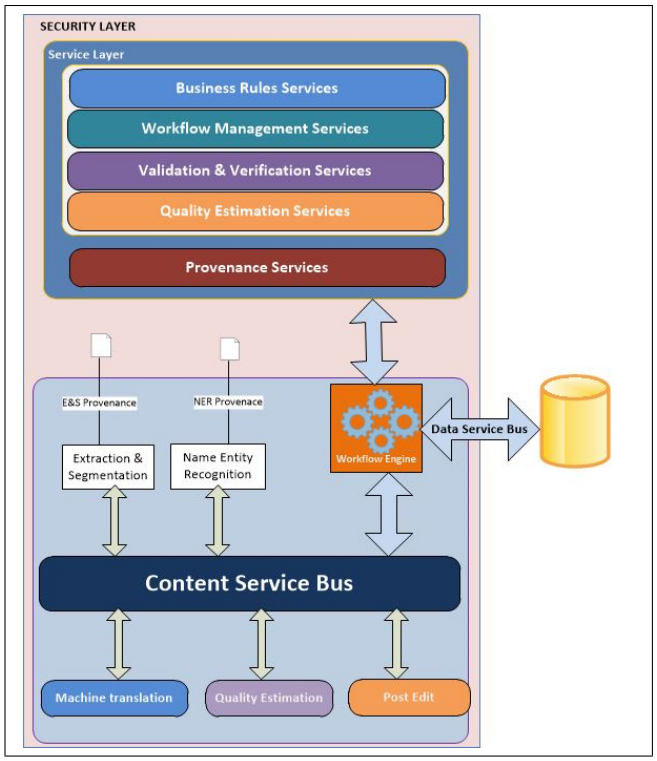


Figure 4. Prototype implementation architecture.

- To share data across multiple component of an application at any time.
- To execute long running operations without overloading the engine, as each component may have individual specific services.
- To use the common content management bus and provisioning infrastructure provided by the Service layer.
- To provide a common policy engine service for the entire platform to reduce the code complexity and improve maintainability.
- To monitor workflow, process executions and record task names, execution durations and parameters use through a provenance service.

The architecture of the system is based on services and standard browser thin clients. We follow the Toolkit script of JavaScript functions that can be used to deploy entire application on a Tomcat web server. The application can be hosted on a Tomcat web server and all services could potentially and be hosted on cloud-based server.

A. Discussion

Explicit variability representation has benefits for the modelling stage. The feature and domain models control the variability, i.e., add dependability to the process design stage. It also allows formal reasoning about families of processes.

Furthermore, the general utility can be demonstrated. The domain and feature models here specifically support domain experts. We have worked with experts in the digital media and language technology space as part of our research centre. Their qualitative feedback, based on expert interviews as the mechanism, confirms the need to provide a mechanism to customise business processes in a domain-specific way. Using the feature model, rule templates can be filled using the different feature aspects guided by the domain model without in-depth modelling expertise. The majority of experts (more than 2/3) in the evaluation have confirmed simplification or significant simplification in process modelling.

In addition, we looked at another process domain to assess the transferability of the solution. In the learning domain, we examined learner interaction with content in a learning technology system [30,31,32]. Again, the need to provide domain expert support to define constraints and rules for these processes became evident. Here, educators act as process modellers and managers, specifically managing the educational content processing as an interactive process between learners, educators and content. Having been involved in the development of learning technology systems for years, tailoring these to specific courses and classes is required.

VI. Related Work

Current open research concerns for process management includes customisation of governance and quality policies and the non-intrusive adaptation of processes to policies. Today, one-size-fits-all service process modelling and deployment techniques exist. However, their inherent structural inflexibility makes constraints difficult to manage, resulting in significant efforts and costs to adapt to individual domains needs.

We discuss related work in the field of constraints and policy definition and adaptive BPEL processes. While a notation such as BPMN is aimed at, there is more work on WS-BPEL in our context. Work can be distinguished into two categories.

- BPEL process extensions designed to realize platform-independence: Work in [23] and [25] allows BPEL specifications to be extended with fault policies, i.e., rules that deal with erroneous situations. SRRF [13] generates BPEL processes based on defined handling policies. We do not bind domain-specific policies into business processes directly, as this would not allow to support user/domain-specific adaptation adequately.
- Platform-dependent BPEL engines: Dynamo [3] is limited in that BPEL event handlers must be statically embedded into the process prior to deployment (recovery logic is fixed and can only be customised through the event handler). It does not support customisation and adaptation. PAWS [1] extends the ActiveBPEL engine to enact a flexible process that can change behaviour dynamically, according to constraints.

Furthermore, process-centricity is a concern. Recently, business-processes-as-a-service (BPaaS) is discussed. While not addressed here as a cloud technology specifically, this perspective needs to be further complemented by an architectural style for its implementation [20].

We have proposed a classification of several quality and governance constraints elsewhere [30]: authorisation, accountability, workflow governance and quality. This takes the BPMN constraints extensions [22,23] into account that suggest containment, authorisation and resource assignment as categories into account, but realises these in a less intrusive process adaptation solution.

 ISBN: 978-1-61208-438-1

The DSRL is a combination of rules and BPMN. Moreover, DSLR process based on BPMN and ECA rules is the main focus on the operational part of the DSRL system (i.e., to check conditions and perform actions based on an event of a BPMN process). There is no need for a general purpose language in a DSLR, though aspects are present in the process language. [33,34,35] discuss business process variability, though primarily from a structural customisation perspective. However, [33] also uses an ontology-based support infrastructure.

VII. Conclusion

In presenting a variability and feature-oriented development approach for a domain-specific rule language for business process constraints, we have added adaptivity to process modelling. We can provide domain experts with a set of structured variation mechanisms for the specification, processing and management of process rules as well as managing frequency changes of business processes along the variability scheme at for notations like BPMN. The novelty of our variability approach is a focus on process constraints and their rule-based management, advancing on structural variability.

Cloud-based business processes-as-a-service (BPaaS) as an emerging trend signifies the need to adapt resources such as processes to different consumer needs (called customisation of multi-tenant resources in the cloud). Furthermore, self-service provisioning of resources also requires non-expert to manage this configuration.

We see the need for further research that focuses on how to adapt the DSRL across different domains and how to convert conceptual models into generic domain-specific rule language which are applicable to other domains. So far, this translation is semi-automatic, but shall be improved with a system that learns from existing rules and domain models, driven by the feature approach, and to result in an automated DSRL generation.

Acknowledgment

This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/CE/I1142 as part of the Centre for Global Intelligent Content (www.cngl.ie) at DCU.

References

[1] Ö. Tanrver and S. Bilgen, "A framework for reviewing domain specific conceptual models," CompStand & Interf, vol. 33, pp. 448-464, 2011.

[2] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," ACM computing surveys, vol. 37:316-344, 2005.

[3] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," DTIC. 1990.

[4] A. Van Deursen and P. Klint, "Domain-specific language design requires feature descriptions," Jrnl of Comp and Inf technology, vol. 10, pp. 1-17, 2002.

[5] M. Acher, P. Collet, P. Lahire, and R. B. France, "A domain-specific language for managing feature models," in ACM Symp on Applied Computing, 2011, pp. 1333-1340.

[6] C. Pahl and Y. Zhu, "A semantical framework for the orchestration and choreography of web services," Electronic Notes in Theoretical Computer Science, vol. 151(2), pp. 3-18, 2006.

[7] C. Pahl, "Semantic model-driven architecting of service-based software systems," Information and Software Technology, vol. 49(8), pp. 838-850, 2007.

[8] C. Pahl, "An ontology for software component matching," International Journal on Software Tools for Technology Transfer, vol 9(2), pp. 169-178, 2007.

[9] M.X. Wang, K.Y. Bandara, and C. Pahl, "Integrated constraint violation handling for dynamic service composition," IEEE International Conference on Services Computing, 2009, pp. 168-175.

[10] Y.-J. Hu, C.-L. Yeh, and W. Laun, "Challenges for rule systems on the web," Rule Interchange and Applications, 2009, pp. 4-16.

[11] A. Paschke, H. Boley, Z. Zhao, K. Teymourian, and T. Athan, "Reaction RuleML 1.0" in Rules on the Web: Research and Applications, 2012, pp. 100-119.

[12] H. Boley, A. Paschke, and O. Shafiq, "RuleML 1.0: the overarching specification of web rules," Lecture Notes in Computer Science. 6403, 162-178, 2010.

[13] T. Soininen and I. Niemel, "Developing a declarative rule language for applications in product configuration," in practical aspects of declarative languages, ed: Springer, 1998, pp. 305-319.

[14] D. Curry, H. Debar, and B. Feinstein, "Intrusion detection message exchange format data model and extensible markup language (xml) document type definition," IDWG, 2002.

[15] K. Williams, M. Brundage, P. Dengler, J. Gabriel, A. Hoskinson, M. R. Kay, et al., Professional XML databases: Wrox Press, 2000.

[16] E. Wilde and D. Lowe, XPath, XLink, XPointer, and XML: A practical guide to Web hyperlinking and transclusion, 2002.

[17] L. Wood, V. Apparao, L. Cable, M. Champion, M. Davis, J. Kesselman, et al., "Document object model (DOM) specification," W3C recommendation, 1998.

[18] E. R. Harold, Processing XML with Java: Addison-Wesley, 2002.

[19] R. Mohan, M. A. Cohen, and J. Schiefer, "A state machine based approach for a process driven development of web-applications," in Advanced Information Systems Engineering, 2002, pp. 52-66.

[20] S. Van Langenhove, "Towards the correctness of software behavior in uml: A model checking approach based on slicing," Ghent Univ, 2006.

[21] P.-Y. Schobbens, P. Heymans, J.-C. Trigaux, and Y. Bontemps, "Generic semantics of feature diagrams," Computer Networks, vol. 51, pp. 456-479, 2/7/ 2007.

[22] K. C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, "FORM: A feature-oriented reuse method with domain-specific reference architectures," Annals of Software Engineering, vol. 5, pp. 143-168, 1998.

[23] M. L. Griss, J. Favaro, and M. d'Alessandro, "Integrating feature modeling with the RSEB," in Intl Conf Software Reuse, 1998, pp. 76-85.

[24] K. Czarnecki and U. W. Eisenecker, "Generative programming," 2000.

[25] D. Beuche, "Modeling and building software product lines with pure variants," in Intl Software Product Line Conference-Volume 2, 2012, pp. 255-255.

[26] D. Benavides, S. Segura, P. Trinidad, A. R. Corts, "FAMA: Tooling a framework for the automated analysis of feature models," VaMoS, 2007.

[27] M. Antkiewicz and K. Czarnecki, "FeaturePlugin: feature modeling plug-in for Eclipse," Workshop on Eclipse Techn, 2004, pp. 67-72.

[28] A. Classen, Q. Boucher, and P. Heymans, "A text-based approach to feature modelling: Syntax and semantics of TVL," Science of Computer Programming, vol. 76, pp. 1130-1143, 2011.

[29] A. v. Deursen, P. Klint, and J. Visser, "Domain-specific languages: an annotated bibliography," SIGPLAN Not., vol. 35, pp. 26-36, 2000

[30] C. Pahl and N. Mani. Managing Quality Constraints in Technology-managed Learning Content Processes. In: EdMedia'2014 Conference on Educational Media and Technology. 2014

[31] S. Murray, J. Ryan, C. Pahl. A tool-mediated cognitive apprenticeship approach for a computer engineering course. 3rd IEEE Conference on Advanced Learning Technologies, 2003.

[32] X. Lei, C. Pahl, and D. Donnellan, "An evaluation technique for content interaction in web-based teaching and learning environments." The 3rd IEEE International Conference on Advanced Learning Technologies 2003, IEEE, 2003.

[33] M.X. Wang, K.Y. Bandara, and C. Pahl, "Process as a service distributed multi-tenant policy-based process runtime governance." IEEE International Conference on Services Computing (SCC 2010), IEEE, 2010.

[34] Y. Huang, Z. Feng, K. He, Y. Huang: Ontology-based configuration for service-based business process model. In: IEEE SCC, pp. 296303. 2013

[35] N. Assy, W. Gaaloul, B. Defude: Mining configurable process fragments for business process design. DESRIST. LNCS 8463, pp. 209224. 2014

 ISBN: 978-1-61208-438-1

Several Issues on the Model Interchange Between Model-Driven Software Development Tools

Una Ieva Zusane, Oksana Nikiforova, Konstantins Gusarovs
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
{una.zusane, oksana.nikiforova, konstantins.gusarovs}@rtu.lv

Abstract — **Models are widely used and are one of the advanced tools of software engineering. There is a necessity to export software models from one software development tool or environment and to import them into another tool or environment, especially actual this task is within the Model-Driven Software Development. Despite of the popular model description standard XML Metadata Interchange (XMI), which can be used to perform the task of the model interchange, several problems according to information loss still are appearing. The research is devoted to the comparison of the tools' abilities to exchange the software model presented in the form of Unified Modeling Language (UML) diagrams with other tools, based on a set of test cases suitable to check the completeness of the model description according to XMI standard. Authors open the discussion about the dependency between tools correspondence to the XMI standard and tool's ability of model interchange.**

Keywords – model interchange; UML diagrams; model-driven software development tool.

I. INTRODUCTION

In software development projects, models are wildly used because they are not only visually easier comprehensible in requirement gathering and design phase, but also model transformations turn them into useable artefacts in implementation phase. Models are usually portrayed as diagrams [1], however they can also be written in a textual modeling language.

Model Driven Software Development (MDSD) uses abstractions provided by models to develop software systems [2]. The development process begins with higher level of abstraction, which is continuously transformed to more detailed levels of abstraction until final system is developed.

During the process there may be a need for model interchange between modeling tools. One scenario is that computation independent model may be designed in one modeling tool, and further work on platform specific model should continue in another tool. Another scenario may be a change of modeling tools in the software development lifecycle due to tools' pricing or available options.

XMI [3] is a popular model interchange standard, which is implemented by many modeling tools. XMI was developed by Object Management Group to improve model interchange abilities between different modeling tools. UML [4] models that are portrayed visually in MDSD support tools, can be converted to text conforming to XMI standard.

In an ideal world model, interchange process should be straight forward, if two tools use the same standard for model interchange. The standard could define precise requirements for interchangeable metadata structures, so that there would be little or no variation for possible. This would provide foundation for errorless model interchange process. However, often there is still a loss of data during the model interchange process. This is caused by different interpretations of the XMI standard, as well as tools' wish to extend XMI with model layout information and different other extensions.

The goal of the research is to evaluate whether there is a dependency between the amount of warnings and errors discovered in MDSD support tools exported XMI file and a modeling tools' XMI model interchange ability. Research consists of two parts. Firstly, tools' ability to export files conforming to XMI standard is evaluated. Secondly, models for three test cases are practically exchanged between tools.

The rest of the paper is structured as follows. Section II describes the importance of model interchange in the MDSD process. In Section III, the National Institute of Standards and Technology (NIST) XMI validation tool is considered as a way to determine the quality of MDSD tools' exported XMI model. The next Section analyses the results of practical model interchange between the modeling tools. Related work is considered in Section V. In the conclusion Section, the results of the research are summarized and the directions of future research are suggested.

II. THE TASK OF THE MODEL INTERCHANGE WITHIN THE MODEL-DRIVEN SOFTWARE DEVELOPMENT

Model is an integral part of MDSD [2], because it can portray different levels of abstraction [1]. MDSD support tools need to have a reliable model interchange ability in order to preserve their users. If a tool cannot cooperate with other development tools, users will have to do a lot of unnecessary manual work, which may lead them to choose another tool for modeling.

Models usually are portrayed visually and saved in MDSD support tools in different formats, which depend on the technology used in the building process of a tool. Some modeling tools can generate XMI files for models, which mean transforming visual models to text. Other tools have an ability to import XMI files, transforming a model from a text file to tools native form of a model. This process is restricted by modeling standards (e.g., UML) and XMI standard rules.

 ISBN: 978-1-61208-438-1

XMI standard provides a set of rules how to write a models' metadata information in Extensible Markup Language (XML) [3]. XML was introduced in 1996 by World Wide Web Consortium [5] with a goal of simplifying data exchange process between different software tools.

In research, three MDSD support tools are reviewed in order to evaluate their ability of model interchange – Enterprise Architect [6], Magic Draw [7] and Modelio [8]. These tools were selected by Model Interchange Working Group in 2011 as ones to be evaluated by Model Interchange Tests [10].

Enterprise Architect is developed by Sparx Systems and has more than 350 000 users in 160 countries [6]. It is a visual modeling tool, which is based on Model Driven Architecture approach developed by Object Management Group. There are numerous modeling standards available in this tool, for example, UML, Business Process Model and Notation (BPMN) and System Modeling Language (SysML). The user interface in Enterprise Architect is user friendly and intuitive, which boosts the usage productivity. The current edition of Enterprise Architect is 12.

Magic Draw is a modeling tool developed by No Magic [7] in order to support object-oriented systems analysis and design. The tool incorporates such industry standards as UML, SysML and BPMN. Magic Draw supports code generation from models to different programming languages (Java, C++, C# and CORBA IDL). The current edition of Magic Draw is 18.1.

Modelio is an open source modeling tool developed by Modeliosoft [8]. The tool is designed for business and system analysts, as well as software developers. Tool consists of modules that can be added to default version of the tool according to user needs. The tool supports code generation from models to Java language. The current edition of Modelio is 3.3.1.

It is possible to export models in XMI format files from Enterprise Architect and Modelio. On the other hand, Magic Draw can export Models only in XML file. This means that all three exported models are written in XML language and more or less conforms to XMI standard. The model interchange problems arise when tool A does not have appropriate transformation rules for XML structures, which are used in tool B.

As an example of differences in tools' interpretation of XMI standard and the way of generating models for interchange purposes, we will look at a small example. In an UML class there is an attribute with name "attribute 1". It is type "Boolean" and can take values from 0 to 1.

In the right part of Figure 1, a fragment of class diagram export from Enterprise Architect can be seen. In addition to previously mentioned characteristics attribute 1 has some more metadata, which are included in Enterprise Architect models.

In the left part of Figure 1 the same fragment of attribute1 exported from Modelio can be seen. This fragment is considerably shorter, as Modelio by default doesn't create as many additional characteristics for an attribute. It is also unclear what the value restriction for this attribute is – this will be considered as a difference between the uploaded XMI file and the "valid XMI" for the test case by NIST validation tool.

Magic Draw exported a file with an extension .xml for class diagram. This tool wildly uses extensions, which make the text form of attribute1 in Figure 2 differ even more from other discussed tools. However, when NIST validation tool compares Magic Draw XML's canonical XMI form to "valid XMI" in Section III the results are good and the amount of discovered validation errors is low in comparison with other tools.

```
<ownedAttribute xmi:type="uml:Property"
xmi:id="EAID_F882A2CB_D820_4476_93B6_B9FFBAF3C03D"
name="attribute1"
visibility="public"
isStatic="false"
isReadOnly="false"
isDerived="false"
isOrdered="false"
isUnique="true"
isDerivedUnion="false">
    <lowerValue xmi:type="uml:LiteralInteger"
    xmi:id="EAID_LI000001_D820_4476_93B6_B9FFBAF3C03D"
    value="0"/>
    <upperValue xmi:type="uml:LiteralInteger"
    xmi:id="EAID_LI000002_D820_4476_93B6_B9FFBAF3C03D"
    value="1"/>
    <type xmi:idref="EAJava_Boolean"/>
</ownedAttribute>
```

```
<ownedAttribute xmi:id="_STUg1-AzEeSVErzShSrkBw"
name="attribute1"
visibility="public">
  <type xmi:type="uml:PrimitiveType"
  href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#Boolean"/>
  <lowerValue xmi:type="uml:LiteralInteger"
  xmi:id="_STUg2OAzEeSVErzShSrkBw"/>
</ownedAttribute>
```

Figure 1. Comparison of XMI atribute1 in Enterprise Architect and Modelio

 ISBN: 978-1-61208-438-1

```
<ownedAttribute xmi:type='uml:Property' xmi:id='_16_5beta2_f00036a_1235745832354_221485_480'
name='attribute1' visibility='public'>
    <type href='http://www.omg.org/spec/UML/20131001/PrimitiveTypes.xmi#Boolean'>
        <xmi:Extension extender='MagicDraw UML 18.1'>
            <referenceExtension referentPath='UML Standard Profile::UML2 Metamodel::PrimitiveTypes::Boolean'
            referentType='PrimitiveType'/>
        </xmi:Extension>
    </type>
    <lowerValue xmi:type='uml:LiteralInteger' xmi:id='_16_5beta2_f00036a_1235745946796_397652_504'/>
    <xmi:Extension extender='MagicDraw UML 18.1'>
        <modelExtension>
            <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='_16_5beta2_f00036a_1235745946796_810458_505'
            value='1'/>
        </modelExtension>
    </xmi:Extension>
</ownedAttribute>
```

Figure 2. XMI atribute1 in Magic Draw

All three MDSD support tools discussed in this paper have different ways of writing an attribute from a class diagram in XML language. When it comes to more complicated parts of models, these differences become increasingly important in the context of model interchange.

III. NIST VALIDATION TOOL

In 2009, Object Management group announced the creation of Model Interchange Working Group (MIWG) [11]. MIWG has created a test suit, which consists of 40 test cases [9]. The test suit allows demonstrating model interchange abilities of several modeling tools. 25 of the tests are defined for UML 2.3 standard. Each test case consists of one or more diagrams and according XMI file, which conforms to XMI standard and is considered as a "valid XMI" for the model. This XMI file is used in the validation process, when the exported XMI files from modeling tools are compared to it.

USA National Institute of Standards and Technology (NIST) [12] has developed a validation tool that can validate an XMI file exported from a modeling tool against the "valid XMI" for a chosen test case. XMI is compared in its canonical form. There are various ways how model can be represented in XMI, which all conform to XMI standard [9]. Canonical XMI has additional points in its specification that eliminates variation. There is only one way in which a model can be correctly represented in the canonical form. Usage of canonical XMI makes it possible to compare two XMI models expressed in it to find the differences. No tools used in the research exports canonical XMI form directly. NIST validation tool converts uploaded XMI files to their canonical form before comparing them to the "valid XMI".

After the validation of an XMI file the summary or results is displayed to the user. In the heading there is information about XMI file: XMI version, object count in the XMI file, used meta-model. It is followed by a list of warnings, which arises when XMI does not fully conform to the Object management groups' developed standard. Warnings may cause problems with model interchange, because the importing modeling tool may not interpret these parts of XMI correctly.

There are two parts of validation errors discovered by the NIST validation tool [12]:

- General errors;
- Differences between the uploaded XMI file and the "valid XMI" for the test case.

If an XMI that is independent from all test cases provided by MIWG is validated by NIST validation tool, only general errors will be displayed.

In order to see differences in the tools' ability to export models, we compared tools in two dimensions.

Firstly, there are several XMI files available from MIWG interchange tests in 2011 [10]. They were exported from MDSD support tools described in the previous Section. All the tools have developed new versions since then, so it is possible to compare the older version of a tool with the new one. Version numbers for tools described in this paper are shown in Table 1.

TABLE I. MODELING TOOL VERSIONS

	Enterprise Architect	MagicDraw	Modelio
Year 2011	9.1	17.0	2.4.19
Year 2015	12.0	18.1	3.3.1

Secondly, all the tools have exported models from the same test cases. This gives the grounds to compare the number of validation errors discovered by the NIST validation tool between the different MDSD support tools.

The comparison of tools in this Section uses three test cases for UML diagrams: class diagram [13], activity diagram [14] and use case diagram [15]. The choice of test cases covers both behavioral and structural UML diagrams.

Class diagrams describe structure of a system showing objects used in a system as classes. Each class can have attributes (characteristics of an object) and methods (actions that an object can do). Classes are linked with each other with different relationships, e.g., association and generalization.

Activity diagrams are used for business process modeling. They display the sequence of activities in a workflow and decisions resulting from activities.

Use case diagrams show user interaction with the system. Users are portrayed as actors and they are connected to use cases by using different links to specify the relation.

Class diagram is the first test case to be analyzed. The comparison of MDSD support tools and data from the tools'

 ISBN: 978-1-61208-438-1

versions from years 2011 and 2015 is shown in Figure 3. The amount of validation errors discovered by the NIST validation tool for the class diagram is displayed there.

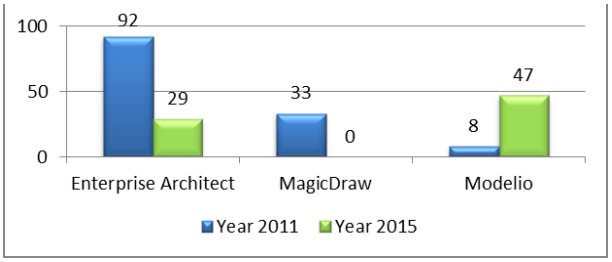


Figure 3. Comparison of validation errors in the class diagram

In the 12.0 version of Enterprise Architect the exported XMI conforms better with the XMI standard than the version 9.1. The amount of validation errors has decreased three times. In the Magic Draw version 17.0 there are 33 validation errors, but in the version 18.1 NIST validation tool did not discover any errors. In Modelio the trend is reversed. In the version 2.4.19 there were 8 validation errors, but in the version 3.3.1 the NIST validation tool discovered 47 validation errors in the exported XMI of the class diagram test case. When making a comparison between different tools, the best in class diagram test case is Magic Draw, which is followed by Enterprise Architect and Modelio. The amounts of validation errors in these tools are increasingly higher.

Activity diagram is the second test case to be analyzed. The comparison of MDSD support tools and data from the tools' versions from years 2011 and 2015 can be seen in Figure 4. The amount of validation errors discovered by the NIST validation tool for the activity diagram is displayed in Figure 4.

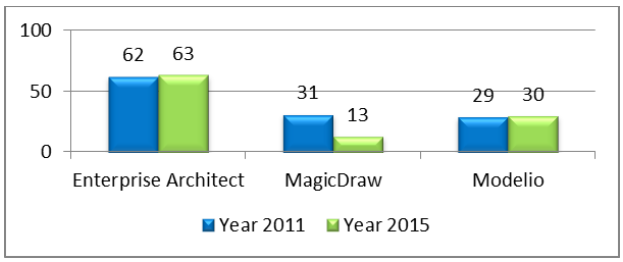


Figure 4. Comparison of validation errors in the activity diagram

When comparing older and newer versions of Enterprise Architect an increase by one validation error can be seen in the version 12.0. Magic draw in newer version 18.1 has 18 validation errors less than version 17.0. Modelio, similarly as Enterprise Architect, in the version 2.4.19 has one validation error more than there was in the version 3.3.1. In the export of activity diagrams Magic Draw has the best results with 13 validation errors discovered in the current tool's version. Modelio has approximately three times more validation errors than Magic Draw, but the highest amount of validation errors belongs to Enterprise Architect.

Use case diagram is the third test case to be analyzed. The comparison of MDSD support tools and data from the tools' versions from years 2011 and 2015 can be seen in Figure 5. The amount of validation errors discovered by the NIST validation tool for the use case diagram is displayed in Figure 5.

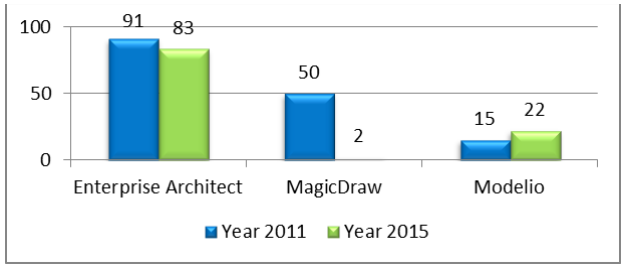


Figure 5. Comparison of validation errors in the use case diagram

The use case diagram XMI file, which is exported from Enterprise Architect version 12.0, has 83 validation errors. That is by 8 validation errors less than Enterprise Architect version 9.1. There is even better improvement in Magic Draw – the amount of validation errors from 50 in version 17.0 has decreased to only 2 in version 18.1. For Modelio, similarly as in the case of class and activity diagrams, an increase in the amount of validation errors for the newer version 3.3.1. can be seen. Magic Draw with its 2 validation errors has the lowest amount of errors for the use case diagram. It is followed by Modelio (22 validation errors) and Enterprise Architect (83 validation errors).

There were various validation errors discovered by NIST validation tool. In the error messages user uploaded XMI file is referenced as "User.xmi" and preloaded XMI for the test case is referenced as "Valid.xmi". The most frequent validation errors were:

- User.xmi is missing an element present in Valid.xmi;
- User.xmi contains an element not present in Valid.xmi;
- An object property value in User.xmi differs from that of Valid.xmi, for example in User.xmi class visibility is defined as "Public", but in Valid.xmi the value is null;
- User object missing a value specified in Valid.xmi.

The highest amount of general errors was about the serialization of a default value.

The summary of all the amounts of validation errors from three test cases for each tool is shown in Figure 6. Summary is made for the tool versions in year 2015.

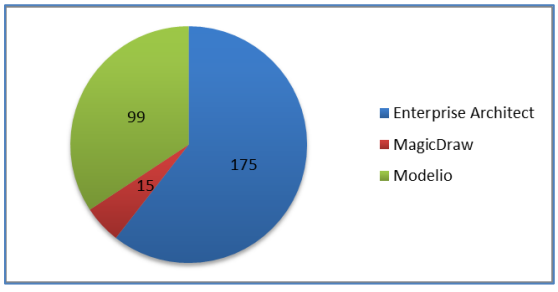


Figure 6. Summary of model validation errors

Enterprise Architect has 175 validation errors, which adds up to 61% from the total amount of validation errors. The majority of validation errors were about the serialization of a

default value. Enterprise Architect specifies the visibility of a public class, where in XMI it is considered a default value for class visibility and should not be specified.

Modelio has half the amount of validation errors. Modelio has a mentionable trend to become less conformant with the XMI standard in the newer version. In all test cases, the amount of validation errors for Modelio version 3.3.1 was higher than for version 2.4.19.

In each test case Magic Draw had the lowest amount of validation errors. Only 5% of the total amount of validation errors was created by Magic Draw.

IV. RESULTS OF THE MODEL INTERCHANGE BETWEEN THE TOOLS

In order to evaluate, whether a model exported form one of described tools can be used in other tools, we tested model interchange practically. In a perfect scenario model interchange should provide a possibility to export a model from one tool and import model in another tool without losing any elements, links and layout.

For each test case analyzed in Section III we practiced model interchange between the described tools and evaluated it according to these criteria:

0 points – model from one tool cannot be imported in another tool;

1 point – model can be imported from tool A into tool B, but it is missing some elements or links;

2 points - model can be imported from tool A into tool B and it has all elements and links;

3 points - model can be imported from tool A into tool B and it has all elements, links and layout.

Results for each test case are displayed in the tables below. Tools named in listed exported diagrams that were imported in tools listed in rows.

TABLE II. CLASS DIAGRAM MODEL INTERCHANGE

To \ From	Enterprise Architect	Magic-Draw	Modelio
Enterprise Architect	X	3	2
MagicDraw	2	X	2
Modelio	1	0	X

Model interchange results for class diagram are shown in Table 2. Modelio was missing some elements in the diagram exported from Enterprise Architect and could not import model from Magic Draw at all. Enterprise Architect could retrieve model layout exported by Magic Draw.

TABLE III. ACTIVITY DIAGRAM MODEL INTERCHANGE

To \ From	Enterprise Architect	Magic-Draw	Modelio
Enterprise Architect	X	3	2
MagicDraw	1	X	2
Modelio	2	0	X

Model interchange results for activity diagram are shown in Table 3. Model did not have all the elements and links in interchange between Enterprise Architect and Magic Draw. Other tools received complete model from Modelio, but did not get the layout.

TABLE IV. USE CASE DIAGRAM MODEL INTERCHANGE

To \ From	Enterprise Architect	Magic-Draw	Modelio
Enterprise Architect	X	3	2
MagicDraw	2	X	2
Modelio	2	0	X

Model interchange results for use case diagram are shown in Table 4. All model interchanges that were functional transported complete models from one tool to another. The only interchange that did not work was from Magic Draw to Modelio.

All the obtained points for both import and export of three test case models are summarized in Figure 7.

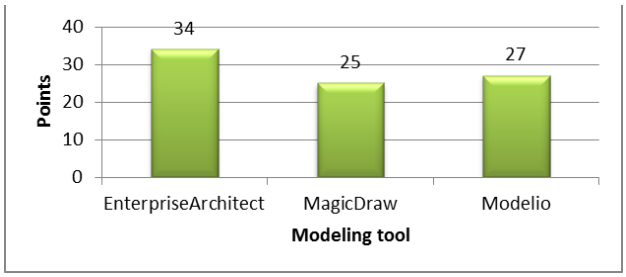


Figure 7. Comparison of practical model interchange

According to previously raised criteria Enterprise Architect has the best ability of model interchange with other modeling tools used in this research. Tools option to import files in XMI, as well as XML formats and ability to take model layout is an advantage.

Modelio has 7 points less than Enterprise Architect. Modelio is best evaluated for the ability to export a model, which can be imported in all other tools with high accuracy.

Modelio is followed by Magic Draw. Tools biggest flaw was its inability to import models layout. Magic Draw offers its users a variety of automatic layout options for models, during practical tests it was recognized that it was not enough. The automatic layout option did not work for use case diagram.

V. CONCLUSION

In this paper the UML model interchange capabilities of three modeling tools were analyzed. The tools are: Enterprise Architect, Magic Draw and Modelio. Three test cases designed by MIWG were used: class diagram, activity diagram and use case diagram.

With the NIST validation tool the largest amount of validation errors were discovered in the XMI files of Enterprise Architect, but the smallest amount of validation

 ISBN: 978-1-61208-438-1

errors were in Magic Draw exported models. When analyzing the trends of development from older to newer versions of tools it can be seen that there is improvement in the amount of validation errors in Enterprise Architect and Modelio. Both tools in year 2015 have less validation errors than they had in year 2011.

In practical model interchange Enterprise Architect is recognized as the most precise of the analyzed tools. It is followed by Modelio and Magic Draw. This result seems to be counterintuitive: Enterprise Architect has the highest amount of validation errors discovered by NIST validation tool, yet it has the best model interchange ability. This could be explained by the tools import abilities, which cannot be evaluated by NIST validation tool. It is also evident that not all validation errors have negative impact on tools ability of model interchange.

In conclusion, the dependency between the amount of XMI validation errors and tools' practical model interchange ability is not evident. The amount of XMI validation errors discovered by NIST validation tool is not enough to determine, whether a MDSD support tool will have good model interchange ability.

One explanation for this result is that only the quality of exported XMI files can be tested by NIST validation tool. Unfortunately, a good conformance to XMI standard does not insure that other modeling tools will import the file successfully. When testing a tools ability of model interchange, both the quality of the export XMI and import to other tools should be examined.

The research can be continued in two directions. Firstly, more MDSD support tools can be compared using the same test cases for UML diagrams. Secondly, the validation errors discovered by NIST validation tool can be analyzed in order to determine, which have significant impact on model interchange.

ACKNOWLEDGMENT

The research presented in the paper is supported by Latvian Council of Science, No. 342/2012 "Development of Models and Methods Based on Distributed Artificial Intelligence, Knowledge Management and Advanced Web Technologies".

REFERENCES

[1] T. Kuhne, What is a Model? Internat. Begegnungs-und Forschungszentrum für Informatik. 2005

[2] S. Beydeda, M. Book, and V. Gruhn, Model-Driven Software Development, Springer, 2005.

[3] XML Metadata Interchange (XMI) Specification. Available: http://www.omg.org/spec/XMI/ [retrieved: July, 2015].

[4] Unified Modeling Language (UML) Resource Page. Available: http://www.uml.org/ [retrieved: July, 2015].

[5] Extensible Markup Language (XML) 1.0 (Fifth Edition). Available: http://www.w3.org/TR/2008/REC-xml-20081126/, [retrieved: July, 2015].

[6] Enterprise Architect. Available: http://www.sparxsystems.com/products/ea/ retrieved: July, 2015].

[7] MagicDraw. Available: http://www.nomagic.com/products/magicdraw.html retrieved: July, 2015].

[8] Modelio. Available: https://www.modelio.org/about-modelio/features.html retrieved: July, 2015].

[9] Model Interchange Wiki. Available: http://www.omgwiki.org/model-interchange/doku.php?id= [retrieved: July, 2015].

[10] UML/SysML Tool Vendor Model Interchange Test Case Results Now Available. Available: http://www.omg.org/news/releases/pr2011/12-01-11.htm, retrieved: July, 2015].

[11] S. Covert, OMG Announces Model Interchange Working Group. Available: http://www.omg.org/news/releases/pr2009/07-08-09.htm, retrieved: July, 2015].

[12] NIST XMI validator. Available: http://validator.omg.org/se-interop/tools/validator, retrieved: July, 2015].

[13] Test Case 1 - Simple Class Model. Available: http://www.omgwiki.org/model-interchange/doku.php?id=test_case_1_uml_2.3, retrieved: July, 2015].

[14] Test Case 4 - Simple (fUML) Activity Model. Available: http://www.omgwiki.org/model-interchange/doku.php?id=test_case_4_uml_2.3, retrieved: July, 2015].

[15] Test Case 8 - Use Cases. Available: http://www.omgwiki.org/model-interchange/doku.php?id=test_case_8_uml_2.3, retrieved: July, 2015].

 ISBN: 978-1-61208-438-1

Testing Smart Cities Through an Extensible Testbed

A Testbed Framework For Smart Cities Validations

Guilherme Luiz Mario de Medeiros[1], Felipe Silva Ferraz[1,2], Gutemberg Rodrigues Costa Cavalcante[1]

[1]CESAR
Recife Center for Advanced Studies and System
Recife, Brazil
guicaraciolo@gmail.com
fsf@cesar.org.br
gutembergrcc@gmail.com

[2]Informatics Center
Federal University of Pernambuco
Recife, Brazil
fsf3@cin.ufpe.br

***Abstract*—Urban areas around the world are being crowded and, in some cases, over populated. This leads to a new set of modern life problems, like traffic jams and natural resource depletion. Information technologies, under the alias of smart city or Internet of things, play a very important role helping analyze, understand, and solve these problems. The present work discusses the need for testing on smart cities and Internet of things field of study. Following this line of thought, it goes through how other researchers are testing and validating their projects. Finally, it presents an easy to use and customize way of generating cheap data mass for project validation.**

Keywords-Testbed; Data Mass; Smart Cities; Internet of Things

I. INTRODUCTION

The growing number of citizens in urban areas is creating a new range of modern problems to humankind. Resource distribution, government transparency, security, mobility, and life quality are just a small set of these new challenges [1]. Some of these problems can be seen worldwide. For example, it is easy to think of at least one name of a city suffering from traffic jams [3], which is one of many mobility problems.

In addition, there is a set of invisible problems related to the growth of population on Earth. Urgent problems like pollution and the depletion of natural resources are already a huge concern in the modern society. The number of people in cities, and how those people behave, affects not only that small location on Earth, but also the entire planet [3][4].

Modern human life has a helpful ally on fighting all of those problems. The increasing power of computational devices and the amount of data gathered about, almost, everything, can help researchers and scientists understand how those problems appear and behave and simulate tools and scenarios, trying to find ways to solve or prevent those problems [4].

The main problem of computational tools is the need to test it. Researchers need to ensure their tools are working as expected, under any circumstance. In Computer Science and software engineering, there are different concepts and tools to test hardware and software, which are being used worldwide by the tech industry. Relying on those tools and techniques would give researchers the certainty their solution may be on the right tracks [5][6].

There are different ways to ensure that software works as expected. One of them is by using a testbed. A testbed comprehends a huge set of tools. In this set, there are tools that behave as a real environment, where other software can be executed in it, "thinking" they are executing on the real environment. In addition, in this set of tools, there are data generators, which simulate a real environment, as the previous testbed, but instead of executing the test subject in it, it outputs a dataset from the simulated environment, which can be used as input for the software the researcher wants to test [5][6][7].

The present work introduces a data simulation testbed, which can simulate the desired target area, and return an almost unique data set. More than that, the final product of this research is a framework to construct testbeds for data mass generation to help testing smart cities and internet of things applications. The main reason of this research is to provide to other researchers a simple to use and extend tool.

The current paper is divided on the following sections: Section II describes the concept of a smart city and what researchers are doing with it; Section III shows how other researchers are trying to solve the lack of smart cities testing tools; Section IV presents how the proposed framework was constructed, the idea and architectural decisions behind it's functionalities; Section V displays the framework validation; Section VI holds a final discussion about this work.

II. SMART CITIES

The expansion of urban areas and the growing number of people on those spaces are creating new problems to humankind like resource distribution or service management. Solving these problems is not simple, which is leading scientists into looking for solutions on, but not exclusively, smart cities [8].

The concept of smart city may vary by researcher, university, or book, but the main idea is the same: try to solve urban problems, save governments money and improve life quality by using Information Technologies (software, hardware, networks, and sensors) [9][14]. However, like any

 ISBN: 978-1-61208-438-1

other research field, smart city also needs validation. For this, the best way of validation is gathering real data from urban area and replicate it to test the solution. The main problem is gathering real world information, which can be expensive, for not mentioning administrative barriers. This can delay the development of a good idea for a long time. In addition, after gathering the desired data, it can disprove the project; meaning money and time were wasted. By exposing this thought, it is easy to understand the need for a reliable and cheap data mass for initial validations on smart cities projects to avoid time and money being wasted [9][10].

Nowadays, it is starting to be common to find public data masses and Application Program Interfaces (APIs), provided by governments for researchers. Nevertheless, those data does not show how citizens behave. For security and bureaucratic reasons, most public data masses relate to government controlled resources usage. Even though these data sets can be helpful and insightful, they are not suitable for all researches [9][10].

To circumvent those problems, researchers are developing testbeds based on real data and observations to generate good data masses for project validations. This may not represent the real world behavior, but such low cost approach is helping researchers to test and validate their ideas early on initial stages.

III. RELATED WORK

In this section, there are enumerated some works and researches that identified the same problem, by quickly explaining their approach to solve it.

A. Simulating Smart Cities with Deus

The study introduces a testbed able to create data mass to test wireless communications infrastructure in a smart city. Its data mass is generated by the frequency probability of a series of discrete events and distributing those events along a time period, using a deterministic algorithm. In this testbed, a discrete event is any state change or communication between nodes – which can be citizens or sensors. The proposed testbed can generate a high amount of data in a small period. On the other hand, it does not offer easy ways to configure data generation [11].

B. Smarty City Application Testbed

This work proposes a testbed for data generation through simulating citizens and resource usage. The application developed provides a way to input configuration options and the testbed tries to simulate behaviors and generate data based on the user input. It seems to be a good idea to enable the test of specific scenarios. The proposal cannot be extended, which means other researchers will not be able to add more modules or entities to the tool simulation, being limited to its generation capabilities [7].

C. Smart City – Platform For emergent Phenomena Power System Testbed Simulator

Differently from the two previous testbeds, this proposal is not about generating data to validate smart cities solutions. The paper proposal is to create a virtual environment where Power Grids (Electrical Grids) projects for smart cities projects can be tested. The main idea is to develop the desired solution in this simulator; while it generates usage patterns and unpredictable events (like natural disasters). This represents a good approach for specific problems especially smart systems that may need to take quick measures without human intervention [5].

D. SmartSantander: The Meeting Point Between Future Internet Research and Experimentation and the Smart Cities

Smart Santander [6][12][15] is a testbed for smart city projects running in a real live city. Researchers can implement their projects and analyze how it would behave in the real world. The architecture is based on different Tiers, and the communication between components among them. Each Tier communicates with a capillary network that carries information to application servers. To facilitate the implementation of new solutions, this architecture is based on only four subsystems, with simple interfaces to use and access.

This project is a major milestone in the field of smart cities. It is hard to point downsides on the proposal since they can also represent new reasons for studies, as the scenario is the most suitable for such.

E. IoT Testbed Business Model

The study presented by Silva et al. [15] is a business model for internet of things testbed deployment on a real environment. It is a derivation of the business ideas and commercialization principles behind Smart Santander [6][12][15]. The model considers four main aspects of a business: infrastructure; value proposition; customers; and financials. Finally, it shows how to detect key partners, key activities, cost structures, and revenue streams, so governments, private corporates or universities can try to implement their own “Smart Santander”.

IV. FRAMEWORK FOR DATA GENERATION

The final product of the present paper is a framework to enable other researchers to create their own testbeds. Those testbeds will generate data mass by simulating a virtual city. For this to happen, the study considered two different configuration moments: simulator configuration and execution profiles. The simulator configuration deals with what the final testbed will simulate and is achieved by adding, removing, or modifying simulation modules. Researchers can use those modules to enable different behaviours to their simulated city. On the other hand, the execution profiles modifies how the simulator will behave. While the simulator just shapes a city, the execution profiles describes how much resources the city will consume during its simulation, which can modify all modules behaviours.

 ISBN: 978-1-61208-438-1

The proposed framework is based on three different parts (as shown on Figure 1): data storage, a Web based Guide User Interface (GUI) and the core application. After generating data, the framework produces a single JavaScript Object Notation (JSON) file, which can be used to validate any smart city project.

A. Data Storage

The main goal of the data storage is to hold execution variables and information, removing this responsibility from the core application. This means the testbed being developed does not need to deal with memory swap, indexing or language garbage collector. Also, abbiding to this architecture decision enables the framework to outsource this concern to another process or software, parallelizing memory management and input and output blocking operations.

For this task, the framework relies on MongoDB - a cross platform document-oriented database, adopted by a number of the most popular websites and services. It focuses on giving fast reading and writing operations. To achieve this, it indexes data into hashes on memory. On the other hand, it does not provide the most common relational operations, like JOIN statements or transactions.

B. Web GUI

The core part of the proposed framework can accept user input, as execution profiles, to generate data mass for specific scenarios. To enable this, the framework provides a stand-alone application, which can behave like a HTTP Web server. Then, the application can connect to the Data Storage (previously described), to save new execution profiles or to download JSON files generated by previous executions. To be able to access all of its functionalities, the user just need to connect to it with a common Web Browser.

C. Core Application

The core application is subdivided in two parts: the Execution Manager and the Simulator. Figure 2 shows how the Core Application and all of its parts behave.

1) Execution Manager

The Execution Manager is capable of connecting to the Data Storage, verify if there are execution profiles to simulate and, for each one of them, raise a new instance of the Simulator. In addition, this Manager holds the responsibility to generate JSON files at the end of each simulation execution.

2) Simulator

The simulator exposes two methods for developers, and comes bundled with a set of simple basic modules.

a) Initialization Method

This method exists for importing and initializing user created and basic modules into the Simulator. By default, it imports and initializes all the basic modules that came bundled with the final application.

This method receives as input the current execution profile, which should be used to initialize the modules that execution method will execute.

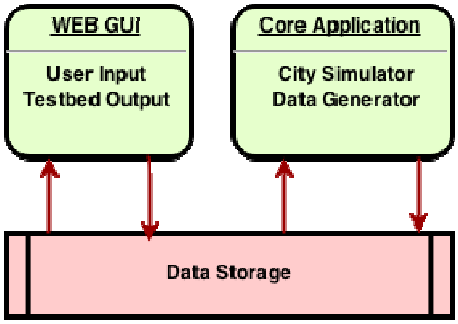


Figure 1. Three Parts Architecture.

b) Execution Method

Three tick events compose the execution method: day changed; hour changed; minute changed. Developers must use at least one of these events to configure and execute these simulation modules. Since the entire execution environment exists on each of these events, everything can be used as input parameter for each module execution method. By doing this, it is easy to inject the current execution date time, or even all the existing citizens, into the module execution.

The goal of splitting the Execution Method into tick events is to facilitate the identification of sets of actions each module may need to perform according to the time of the day. More than that, it helps overhead reduction by avoiding unnecessary method calls. For example, the “traffic module” needs to be simulated each minute, so it can generate data as real as possible. On the other hand, “education access module” can be simulated once a day, may be at the end of the day (day changed event), since it may only need to simulate students score based on its frequency. In addition, adopting this architecture enables modules to have different execution methods for each tick event, improving readability on module’s code.

Notice that, like the real world, time is an incremental value, being orchestrated by the smallest time unit, which is, for the proposed framework architecture, the minute unit. The default real world, and framework, behavior is always increment the smallest unit by one. For the framework, the incremental value can be modified (in the Initialization Method) to make the simulation better suit the target project needs. For example, instead of incrementing by the default value, it could increment the minute unit by ten, reducing the number of minute tick events, decreasing generation time.

 ISBN: 978-1-61208-438-1

Finally, this architecture opted for not using parallel module execution, like threads or sub-process. This decision relates to the passage of time and each module execution state. Using threads or sub-process would force the framework and its modules to implement execution signals to ensure that different modules are still executing the same simulation moment. In addition, since modules can be used as parameters for other modules executions, using parallel processing could damage this concept, since injected modules could change their own execution state in the middle of the execution of the dependant module, making the chain execution fail.

c) Default Basic Modules

The framework comes bundled with three basic modules, which represents basic components of a city. Even though those modules can be used to generate data, their only purpose is to serve for this paper validations.

The bundled modules are:

- The City Manager – capable of generating a city. The size of the output city is based on user input (from the WEB GUI). The final city is randomly generated based on a predefine probability for each created block: 65% chance to be a home block; 25% chance to be a workplace; 1% chance to be a hospital; 1% chance to be a police station; 3% chance to be a school; 5% chance to be a leisure block and 1% chance to be a university block.
- The Citizen Manager – generates citizens for the simulation, gives those homes and jobs, and makes them walk through the city. Jobs and age distribution is based on a set of predefined "citizens profile", where each profile has a chance of service consumption. The Citizen Manager tries to distribute this profiles based on input for service access.
- The Transport Manager – can only be initialized by injecting a city into it. At each iteration, receives a list of citizens that want to move to another location, tries to move them, and remove from this list the citizens that got to their destination.

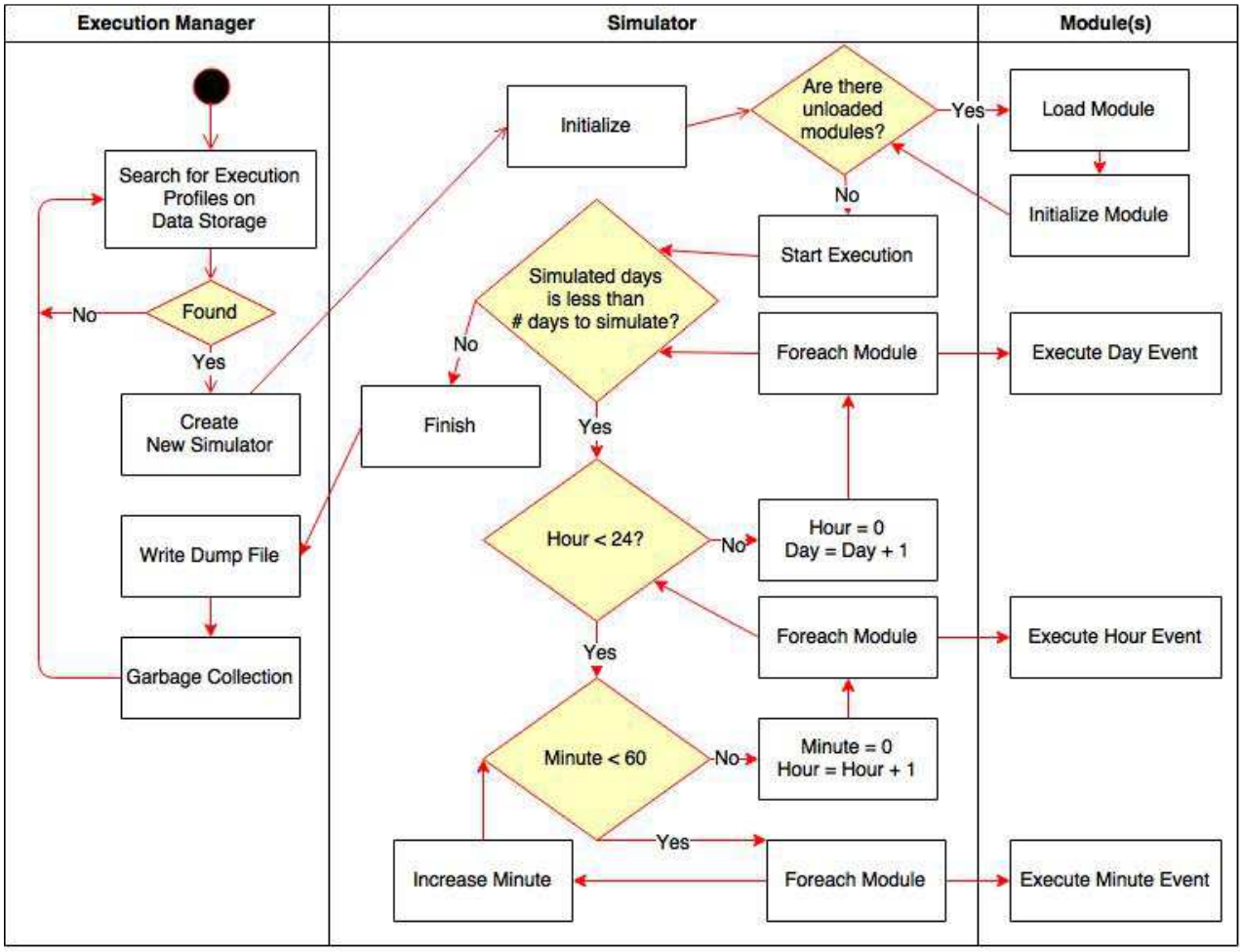


Figure 2. Core Application Workflow.

 ISBN: 978-1-61208-438-1

V. VALIDATION

To validate the framework, three different scenarios were created: a simple village; *Espinheiro*; and *Boa Viagem*. While the village scenario tries to represent a fictional small village, which will help to validate the framework speed for small inputs, both *Espinheiro* and *Boa Viagem* scenarios try to mimic two huge neighborhoods from *Recife* (a city in Brazil). For validation and testing, data about those neighborhoods were gathered from [13]. Since this data source only deals with area size and number of citizens, and the framework needs to have an execution profile in order to work, for service consumption behaviour, random numbers were used. Even though these numbers may not represent real world services usage, they are good enough to validate the framework. The input data for each execution scenario can be found in Table 1. For these validations, the framework's default modules were used. Because of that, since the city generation module, created to validate the framework, only deals with blocks of buildings, for *Espinheiro* and *Boa Viagem* scenarios, a block was considered as having 100m2. Finally, each scenario was executed three times. A summary of each execution can be found on Table 2.

To execute these validations, a Virtual Private Server (VPS) were used, with the following environment specifications:

- One Virtual CPU with 2GHz.
- 1 GB RAM
- 30 GB SSD Hard Drive

TABLE I. VALIDATION SCENARIOS

Input	Scenarios		
	Village	***Espinheiro***	***Boa Viagem***
City Width	10	72	251
City Height	10	100	251
N. of Citzens	2000	10000	130000
N. of Days to Simulate	90	90	90
Total Education Service	35000	30000	130000
Total Health Service	41000	80000	877500
Total Transport Service	280000	2250000	12090000

- Ubuntu 14.04x64
- Python 2.7.6
- MongoDB 3.0.2.

Analyzing the summary output, it is easy to understand the impact the execution profile has on execution time and generated file size. Since for each scenario the input grew on city size, number of citizens and service consumption, the output for them also grew on execution time and generated file size. This means that a testbed generated by the framework is reusable for different city behaviours in the same virtual city, which means one of the two configuration moments, described in section IV, was achieved.

TABLE II. EXECUTION OUTPUT

Output	Village			Espinheiro			Boa Viagem		
	First	***Second***	***Third***	***First***	***Second***	***Third***	***First***	***Second***	***Third***
Execution Time (Seconds)	865.26	832.18	813.12	5479.79	5798.20	5272.86	32308.51	32420.90	32511.08
Data Mass File Size (MB)	63.8	60	59.2	397	396	387	1.7 GB	1.7 GB	1.7 GB
Home Blocks	73	57	66	4560	4636	4648	40826	40766	40802
Work Places Blocks	18	26	20	1857	1785	1822	15603	15704	15651
Hospital Blocks	1	2	1	68	87	62	617	637	694
School Blocks	5	6	2	220	217	210	1946	1813	1976
University Blocks	1	1	2	39	40	30	303	285	301
Police Station Blocks	1	2	1	74	70	56	643	646	647
Leisure Blocks	1	6	8	382	365	372	3063	3150	3109
School Citzens	72	94	103	851	849	831	366	364	364
Graduate Citzens	110	104	102	831	822	854	344	382	382
Gratuate-worker Citzens	112	88	87	826	819	785	356	349	349
Postgrad-worker Citzens	95	103	97	826	844	864	379	350	350
Worker Citzens	955	972	979	6666	6666	6666	64631	64667	64667
Retired Citzens	656	639	632	0	0	0	63924	63888	63888
Avarage Citzens Age	50	49	49	35	35	35	58	58	58

 ISBN: 978-1-61208-438-1

Continuing with the summary analysis, it is also clear how biased is the default module for city generation. Previously, it was mentioned that the default module has a probability for each block creation. Taking the "police station" and "hospital" probability as example, which is a 1% chance, and looking at the results, we can visualize the number of blocks for theses block types is always near 1%. Even though their coordinates may be very different, which may affect traffic related studies, the way blocks are being generated for different size of cities may not represent a city growth behavior.

In addition, the chosen framework architecture may not suit for larger simulations, since it does not deal with parallelism. The effect of this decision can be seen on the execution times presented in Table 2. The smaller scenario took fourteen minutes to process and the biggest scenario took eight hours. However, the proportion between each configuration entry for their execution profiles is not higher than sixty times. Even though execution time may be reduced by using a powerful CPU and more RAM memory, the gain may not be perceived on large simulation scenarios.

Finally, these validations prove that the framework really works, that it can be customized by different service modules, and also show how the execution profiles may affect the execution time and final file size. Even though the default modules work well and may be used for some use cases, they only exist to show how modules work inside the proposed framework and, as this validation shows, it is highly recommended that researchers extend or replace the default modules, or even add more city service modules.

VI. CONCLUSION

The effort of testing a solution is compensated by finding problems as soon as possible. Finding problems on tools in the early stages of a development, means fixing the solution early, when it is easier and cheaper. Also, battle testing the software or hardware being developed reduces the chance of an unknown problem being released worldwide. Even though this is common in the tech industry, the reasoning of using tests can be easily related to Smarty Cities and Internet of Things research.

The result of this paper is a framework for data generation through cities simulation. The generated data can be used in any smart cities or internet of things research. In addition, the proposed tool means a cost reduction, since researchers do not need to waste money on gathering real environment data until it is necessary. As shown in the validation section, researchers can simulate ninety days in only eight hours, which is a precious time saving. However, this time could be decreased if the framework supported parallel processing.

For future works, there is a need to research more on the framework modules. Implementing new modules for specific set of city and world functionalities would be helpful for other researchers. In addition, in the software engineering field, there is a need to find ways of optimizing the time the framework takes to generate data, but keeping it easy to use and customize. It would be great to see researches and projects on Smart Cities and Internet of Things using the testbed framework as one of its tools for project test and validation.

REFERENCES

[1] C. Harrison et al., "Foundations for Smarter Cities," IBM J. Res. and Dev., Jul. 2010, pp. 1-16, ISSN: 0018-8646.

[2] G. Coulson et al., "Flexible Experimentation in Wireless Sensor Networks", in Magazine Communications of the ACM, Volume 55 Issue 1, Jan. 2012, pp. 82-90, doi: 10.1145/2063176.2063198

[3] S. Dirks and M. Keeling, "A vision of smarter cities: How cities can lead the way into a prosperous and sustainable future," IBM Inst. Bus. Value, in Executive Report, June, 2009.

[4] Forrester. *Helping CIOs Understand "Smart City" Initiatives*. [Online]. Available from http://www-935.ibm.com/services/us/cio/pdf/forrester_help_cios_smart_city.pdf 2015.10.28

[5] L. Lugaric, G. S. Member, S. Krajcar, and Z. Simic, "Smart City - Platform for Emergent Phenomena Power System Testbed Simulator," IEEE, Innovative Smart Grid Technologies Conference Europe (ISTG Europe), Oct. 2010, pp. 1–7, doi:10.1109/ISGTEUROPE.2010.5638890.

[6] D. Carboni, A. Pintus, A. Piras, A. Serra, A. Badii, and M. Tiemann, "Scripting a Smart City: The CityScripts Experiment in Santander," 2013 27th Int. Conf. Adv. Inf. Netw. Appl. Work., IEEE, Mar. 2013, pp. 1265–1270, doi:10.1109/WAINA.2013.85.

[7] D. Silva, F. Ferraz, and C. Ferraz, "Smart City Applications TestBed Towards a service based TestBed for smart cities applications," in SOFTENG 2015 : The First International Conference on Advances and Trends in Software Engineering Information, Apr. 2015, pp. 104–107, ISBN: 978-1-61208-449-7.

[8] F. Ferraz, C. Sampaio, and C. Ferraz, "Towards a Smart City Security Model Exploring Smart Cities Elements Based on Nowadays Solutions," ICSEA 2013, The Eight International Conference on Software Engineering Advances, 2013, pp. 546–550.

[9] T. Nam and T. a. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," Proc. 12th Annu. Int. Digit. Gov. Res. Conf. Digit. Gov. Innov. Challenging Times - dg.o '11, Jun. 2011, pp. 282, doi: 10.1145/2037556.2037602.

[10] W. Da Silva el al., "Smart cities software architectures", in Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13, 2013, pp. 1722, doi: 10.1145/2480362.2480688.

[11] M. Picone, M. Amoretti, and F. Zanichelli, "Simulating Smart Cities with DEUS," Proc. Fifth Int. Conf. Simul. Tools Tech., 2012, pp. 172-177, ISBN: 978-1-4503-1510-4.

[12] L. Sanchez et al., "SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities," Future Network and Mobile Summit (FutureNetw), Jun. 2011, pp. 15-17, ISBN: 978-1-4577-0928-9.

[13] Recife Open Data Website. [Online] Available from: http://dados.recife.pe.gov.br/ 2015.09.30

[14] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," IEEE Internet of Things Journal, Vol. 1, No. 1, Feb. 2014, pp. 22-32

[15] E. Silva and P. Maló, "IoT Testbed Business Model," Advances in Internet of Things, 4, 2014, pp. 37-45, doi:10.4236/ait.2014.44006

 ISBN: 978-1-61208-438-1

Implementing the Observer Design Pattern as an Expressive Language Construct

Taher Ahmed Ghaleb, Khalid Aljasser and Musab Al-Turki

Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Emails: {g201106210, aljasser, musab}@kfupm.edu.sa

***Abstract*—Observer is a commonly used design pattern as it carries a lot of reusability and modularity concerns in object-oriented programming and represents a good example of design reuse. Implementing the observer design pattern (and several other design patterns) is known to typically cause several problems such as implementation overhead and traceability. In the literature, several approaches have been proposed to alleviate such problems. However, these approaches only considered the implementation of a specific scenario of the observer pattern, which is concerned with having a single subject with multiple observers. In addition, the code used to implement this pattern was scattered throughout the program, which complicated implementing, tracing and reusing them. In this paper, we: A) provide a systematic classification of all possible scenarios of the observer design pattern and B) introduce a novel approach to implement them using an expressive and easy-to-use construct in Java. The proposed observer construct is built as a language extension using the *abc* extensible compiler. We illustrate through several observer scenarios how the construct significantly simplifies the implementation and improves reusability.**

Keywords–Design Patterns; Aspect-Oriented Programming; Extensible Compiler; Language Extension; Observer Pattern.

I. INTRODUCTION

Object-oriented (OO) design patterns [1] are reusable solutions that reorganize OO programs in a well-structured and reusable design. They originally were implemented using OO features, such as polymorphism and inheritance. After Aspect-oriented (AO) programming languages emerged, researchers started to employ AO constructs to make the implementation more reusable and modular.

Despite the wide range of applications of design patterns, manually implementing them may lead to several problems including most notably implementation overhead, traceability and code reusability [2]. A programmer may be forced to write several classes and methods to achieve trivial behaviors, which leads to a sizable programming overhead, scattering of actions everywhere in the program, and reducing program understandability. Although design patterns make design reusable, the code (or at least part of the code) used to implement them cannot be reused later.

Although the observer pattern has been widely used in practice, a systematic investigation of methods of its implementation considering all the potential observing behaviors was missing in the literature. In particular, the only implemented scenario of the observer pattern in the literature is the one having a single subject with multiple observers, where the association of observers to subjects is subject-driven. For instance, implementations of the observer design pattern in [3] and [4] were illustrated using the example of having $Line$, $Point$ and $Screen$ classes, where the observing protocol is implemented in a way that a single subject can have a list of observers. This particular example actually shows a different case where many subjects (i.e., $Lines$ and $Points$) can be observed by a single observer (i.e., $Screen$). Another issue of conventional implementations of the observer pattern is concerned with the indirect way of implementing the pattern. In other words, programmers in such approaches cannot deal with the pattern as a recognizable unit in programs. Instead, they are required to build an observing protocol and apply it to each instance interested in observing a particular subject, which leads to increased dependencies in the programmer's code.

Motivated by this, we aim in this paper to address these issues while making two main contributions. First, we systematically study and classify the possible scenarios of applying the observer design pattern. This classification is essential for gaining a comprehensive understanding of the structure, design and usage of the observer pattern. Second, we introduce a novel approach to implement the observer design pattern (with all its possible scenarios) as an identifiable language construct. This approach is implemented as a language extension providing a very expressive and easy-to-use observer construct. The implementation of the observer pattern in this approach becomes more explicit and is significantly simplified as shown in the typical examples presented in the paper. Consequently, this implementation approach promotes code correctness by reducing chances of making programming errors (both in the implementation of the pattern and the code using the pattern), resulting in increased productivity, enhanced modularity, and reduced dependencies between modules.

The rest of the paper is organized as follows. Section II describes the observer pattern and presents a systematic classification of its possible scenarios. Section III describes the syntax and semantics of the proposed construct of the observer design pattern and how it can be applied. In Section IV, we discuss the characteristics of our approach and present some potential improvements to be considered in the future. Related work is then presented in Section V. Finally, Section VI concludes the paper and suggests possible future work.

II. SCENARIOS OF THE OBSERVER DESIGN PATTERN

The observer design pattern allows monitoring changes in some components of the program, called subjects, to notify other parts of the program, called observers. It consists of two main components: subjects and observers. In general, the observer pattern may define a many-to-many dependency between subjects and observers, in which changes in the states of subjects cause all their respective dependents (i.e., observers) to be notified and updated automatically.

 ISBN: 978-1-61208-438-1

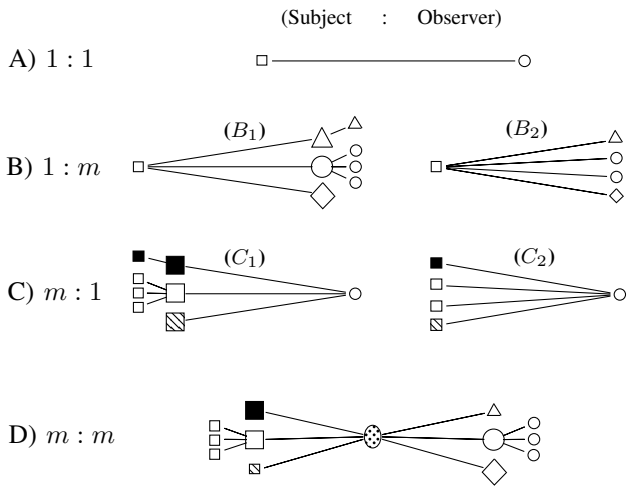


Figure 1. Scenarios of the observer design pattern
[small shape = instance, big shape = class]

Typically, however, the observer pattern represents the case in which a subject maintains a set of observers, and notifies them whenever it has changes in its state [1] (i.e., one subject - many observers). This case is actually limited to one scenario in which the association of observers to subjects is made on basis of subjects. In other words, observing a list of subjects by an observer requires each of these subjects to utilize an individual observing protocol containing a single observer in its list. The proper alternative way to implement such a case would be to have another observing protocol that can associate a list of subjects for any interested observer (i.e., an observer-oriented protocol). Another problem of this implementation is the *instance-level* application in which every instance of an observer class has to explicitly be assigned to the observed subject. This would be better achieved using a *class-level* association of observers to subjects. This means that a subject can be observed by a class, and then all instances of that class will implicitly be assigned to the list of observers of that subject.

Below, we present a systematic classification of the different scenarios of using the observer design pattern.

A. Single Subject - Single Observer

In this case, an observer can only observe a single subject, and the subject can only be observed by one observer as shown in Figure 1(A). This kind of observing is said to be a $1 : 1$ association, where a notification of a state change of the intended subject is sent to the corresponding observer. This scenario is also viable when a certain subject has many attributes, and a certain observer is interested in observing a single attribute of that subject. Therefore, the association of the observer to the subject in this case is also considered as *one-to-one*. This association of a single observer to a single subject can be applied using an *instance-level* observing. However, it can also be applied using a *class-level* observing provided that the observing and observed classes are singletons (i.e., each of them has a single instance).

B. Single Subject - Multiple Observers

This is the common scenario of the observer pattern that describes the case where a single subject can be observed by a set of observers of different types (Figure 1(B)). This means that whenever the subject changes its state, all its dependent observers are notified. For example, when a central database has changes in its data, all dependent applications to this database are notified. In addition, observing a single attribute of a certain subject by many observers is another case of this scenario. This scenario can be applied at two different levels:

1) Class-level: This case happens when a single subject is observed by many observing classes (each with a single instance or multiple instances) as shown in Figure 1(B.B_1). The association of the subject to all corresponding observer classes is said to be $1 : m$. The other case of this scenario occurs when a single subject instance or an attribute of that subject is observed by an observing class with multiple instances as shown in the same sub-figure where the subject is observed by the *circle* class. This leads to the implicit application of the observing logic to all instances of that class to have an association of $1 : m$ as well.

2) Instance-level: Here, a subject instance/attribute can have a list of observing instances (either of the same or different class types) in a $1 : m$ association (demonstrated in Figure 1(B.B_2)). In this case, every instance should explicitly be listed as an observer to the corresponding subject. It should be noted that some instances of the same class may be interested in observing the intended subject while the others may not.

These two levels of association can actually happen together, where a subject can be observed by different class types and at the same time by instances of other classes. Moreover, the *class-level* observing can be applied when all instances of an observing class need to participate in the observing, whereas it is required to apply an *instance-level* observing when only some of the instances are interested in observing that subject.

C. Multiple Subjects - Single Observer

It is common to have one observer that has the responsibility of observing several subjects at the same time. For example, a weather station class may observe different classes for temperature, humidity, wind, etc. As presented in Figure 1(C), this association can be represented as $m : 1$ where the multiple subjects can either be of the same class or different classes. Similar to the previous scenario, *class-level* and *instance-level* observing can be applied in this scenario as shown in Figure 1(C.C_1) and (C.C_2), respectively. Here, an observer instance can observe either a single subject class (all instances of that class are implicitly observed), a list of subject classes, a list of subject instances (from same or different class types), or a set of attributes of a certain subject.

D. Multiple Subjects - Multiple Observers

This scenario encompasses all the previous cases in an $m : n$ association. This kind of association is demonstrated in Figure 1(D) and occurs when several observers intend to observe many subjects. This can also be applied as a *class-level* observing or an *instance-level* observing or both together. When subjects (same or different class types) have more than one attribute to observe, then we might have a combination of several scenarios. An example of this scenario can be described by having a set of class and instance observers interested in observing a class subject, an instance subject and an attribute of a subject.

 ISBN: 978-1-61208-438-1

III. Observer as a Language Construct

In this paper, we propose a novel approach for implementing the observer design pattern as a language extension. This language extension conveys the idea of having an expressive construct that allows explicit application of the pattern using recognizable easy-to-use statements. Actually, such an extension can be built on top of any AO programming language by means of extensible compilers.

In this work, the implementation of the language extension is conducted using the *abc* extensible compiler [5]. *Abc* employs *Polygot* [6] (a Java extensible compiler framework) as a frontend, and extends it with AspectJ constructs. This allows programmers to extend the compiler's syntax and semantics of both: Java and AspectJ. The immediate concrete implementation of the observer construct is based on AO constructs, which is automatically generated using the parameters passed through the construct statements. This implementation is more modular compared with pure OO implementations in Java since it uses the crosscutting facilities provided in AspectJ [7][8]. Modularity in our approach is improved with the use of the high-level and parametrized observer construct that makes using this pattern more expressive and intuitive. At its final stages, *abc* transforms AspectJ Abstract Syntax Tree (AST) into Java AST while preserving the aspect information, and then performs all required weaving with the help of the *Soot* analysis and transformation framework [9] that is used as a backend.

A. Syntax

The observer pattern language construct is designed to be as abstract and modular as it could possibly be while maintaining high accessibility to programmers and users. Moreover, the construct allows applying all possible scenarios of the observer pattern expressively with the least amount of code. Its syntax is defined using the following EBNF notation:

```
<LetObserve>::= "let" <annotated_id_list>
                "observe" <extended_id_list>
                ["exec" <method_invocation>] ";"
```

The observer construct consists of three parts: (1) a list of one or more observers specified by a comma-separated list of class and/or object identifiers given by `<annotated_id_list>`; (2) a list of one or more subjects given by `<extended_id_list>` specified by a comma-separated list of any combination of class and object identifiers and attribute names or even the wildcard (*) to refer to all attributes within the subject to be observed; and finally (3) a single optional notification method given by the `<method_invocation>` non-terminal. Each of the two non-terminals `<annotated_id_list>` and `<extended_id_list>` has its own production rules defined in our extension (as shown below). The `<method_invocation>` and `<name>` non-terminals are already defined in the Java 1.2 parser for CUP [10] employed by *abc*. The production rules that define the non-terminal `<annotated_id_list>` are given as follows:

```
<annotated_id_list>::= <id> {"," <id>}
<id>::= ("class" <name> | <name>)
```

; where the `class` keyword is used to distinguish between class and object identifiers (especially when declared with the same names). The non-terminal `<extended_id_list>` defines an extension to the non-terminal `<id>`. This extension allows programmers to assign subject names, determine certain attributes of them to observe, or use the wildcard ($\star$) to refer to all attributes within a subject to be observed, as follows:

```
<extended_id_list>::= <ext_id> {"," <ext_id>}
<ext_id>::= <id> ["(" ("*" | <attrib_list>) ")"]
<attrib_list>::= <name> {"," <name>}
```

As an example statement that can be generated by this syntax, the following statement:

```
let screen1, class Log observe line1(length), class Point(*);
```

sets up an object $screen1$ and a class Log as observers for changes in length attribute of object $line1$, and any change in state of any object of class $Point$. From now on, we refer to such statements that can be generated by this syntax as '$let - observe - exec$' statements.

In general, the construct can directly support the application of all the scenarios of the observer pattern as described in Section II above (with both: class-level and instance-level observing). In its current implementation, however, scenarios involving mixed usage of *instance-* and *class-level* observing can be specified by multiple separate '$let - observe - exec$' statements, rather than a single statement, which is to be improved upon in future versions of the implementation (See Section IV-C).

B. Application

To show the implementation of the observer construct and how it can be applied, we define three Java classes and several instances of them in Table I: $Line$ and $Point$ as subjects while $Screen$ as an observer. In the $Application$ class, we create some instances of these classes to utilize them in the *instance-level* application of the construct. Some scenarios of the observer pattern require all instances of a class to observe subjects (i.e., *class-level* observing), while some others need every instance to have its own observing logic (i.e., *instance-level* observing). The observer construct provides both *class-* and *instance-level* observing. The general structure of the observer construct is as follows: observers (classes and instances) are placed after the let keyword, subjects (classes and instances) after the $observe$ keyword, and, optionally, the notification method after the $exec$ keyword.

1) Class-level Observing: The *class-level* observing can be applied as follows:

```
let class Screen observe class Line, class Point;      (-1-)
```

In this kind of observing, programmers can indicate that one class is observing a subject class or a set of subject classes. Consequently, all instances of the observing class will be notified when any instance of the subject(s) has state changes. This application shows a case of the *class-level* version of *Multiple Subjects - Multiple Observers* scenario that is applied using only one statement.

2) Instance-level Observing: The observing logic in the *instance-level* version of the observer pattern is accomplished instance-wise. This means that each constructed object of the observing class may observe various subjects with a different number of attributes of each subject. One form of this kind of observing is to observe a single attribute of a single subject, as follows:

```
let screen1 observe line(length) exec resize(length);  (-2-)
```

 ISBN: 978-1-61208-438-1

TABLE I. FOUR JAVA CLASSES: TWO SUBJECTS, AN OBSERVER, AND AN APPLICATION

First Subject Class

```
class Line {
  Color color;
  int length;
  void setLength(int len){
    this.length = len; }

  void setColor(Color c){
    this.color = c; }
}
```

Second Subject Class

```
class Point {
  int x, y;
  void setPos(int x, int y){
    this.x = x;
    this.y = y; }
}
```

Observer Class

```
class Screen {
  public void resize(int len){
    System.out.println("Resizing
                  with the new length: " + len); }
  public void display(String str){
    System.out.println(str); }
}
```

Application

```
Line line = new Line();
Point point = new Point();
Screen screen1, screen2, screen3 = new Screen();
```

This case refers to the *Single Subject - Single Observer* scenario in which the programmer has to specify the observing instance, the subject and the notification method that will receive the change of the state of the specified attribute of the subject and send it directly to the corresponding observer. Another form is to observe multiple attributes of single subject by one observing instance. This form represents the *Multiple Subjects - Single Observer* scenario with the case of observing many attributes of a subject using one statement, as shown in the following application:

```
let screen2 observe line(color,length) exec display;    (-3-)
```

The restriction of this application is that the programmer has to define only one notification method (with a $String$-type parameter) to refresh the observing instance with the state changes of all attributes of the subject. If the programmer did not specify the notification method, the compiler is built to assume that there exist a method called '$display$' in the observing class will do the job.

Last form is to observe multiple subjects with all their attributes using one statement as shown below. This form also represents the *Multiple Subjects - Single Observer* scenario but now with the case of having many subjects with either single or multiple attributes per each. This could be accomplished by either not specifying the attributes at all, or by using the wildcard (*) to refer to all attributes. With respect to specifying the notification method, cases of the previous form also apply here.

```
let screen3 observe line, point(*);                     (-4-)
```

C. Semantics and Code Translation

After parsing '$let - observe - exec$' statements and matching them with the given syntax of the construct, the compiler then moves into other compilation passes that are concerned with the construct semantics. During these passes (with the help of the type system), the compiler starts recognizing class types, instances, attributes and methods used in the construct application by carrying out scoping and type-checking operations. If such checking is passed successfully, the compiler then carries out the code conversion (or rewriting). Otherwise, a semantic exception is generated by the compiler.

1) Variable Scoping: The compiler checks the validity of each element of the observer construct (i.e., classes, instances, attributes and the notification method) to see whether they are not defined or out-of-scope. The compiler in such cases will generate a semantic exception. Another check is conducted when the construct is applied without specifying a notification method. In this case, a programmer has to define a notification method named $display$ in the observing class to be responsible for refreshing it with the changes happened. If such a method is not defined, the compiler will also produce a semantic exception.

2) Type checking: In this process, the compiler is going to pick the class included in the observer construct, and checks its eligibility. For instance, when the programmer uses an observer construct for primitive types, the compiler will check and produce an appropriate alert message showing that only classes or instances can be applied. Also, when programmer use the *instance-level* observing form, then the argument type of the notification method must match the type of the observed attribute. For the case of applying the construct with a default notification method, the compiler would expect programmers to define a method called $display$ in observing class that accepts the changes as a $String$ type.

3) Node Translation and Code Conversion: After achieving all checks successfully, the compiler starts converting LetObserve nodes into their corresponding aspect declaration nodes that the original AspectJ compiler can deal with. This node translation is actually executed through a code conversion pass of the compiler where each '$let - observe - exec$' statement is converted into a specialized aspect that contains the proper crosscutting concerns of the observing statement as shown in Table II.

Every auto-generated aspect is assigned a name of the form '$ObserverProtocol\_\#$', where the hash symbol refers to a sequence number that will be assigned for each auto-generated observing aspect. The newly generated node (i.e., the aspect declaration) is created outside the class that contains the application of the observer construct. Indeed, aspects generated for *class-level* observing purposes have a different implementation style from the ones used for *instance-level*.

- ***Class-Level* Observing**: As shown in Table II.A, an aspect is generated for the '$let - observe - exec$' statement (1). This aspect implements the observing logic for all instances of the supplied observer class in the statement. Therefore, a list of observers (Line 3) is employed to hold a reference copy for every newly created object of that observer class. Object construction joinpoints are crosscutted using the pointcut declared in Lines 5-6 and are advised in Lines 8-10. Whenever a subject has changes on its associated attributes, the $subjectChange$ pointcut (declared in Lines 12-14) will be executed. Consequently, every instance of that observing class will be notified (this task is accomplished by the advice declared in Lines 16-24). After a successful generation of the desired aspects, the compiler replaces '$let - observe - exec$' statements by empty statements (i.e., semicolons ';').
- ***Instance-Level* Observing**: In *instance-level* observing, an aspect is also generated for the '$let-observe-exec$' statement (2) as shown in Table II.B. This aspect

TABLE II. AUTO-GENERATED ASPECT FOR THE OBSERVER PATTERN CONSTRUCT

A. *Class-level* Observing

```
1  protected privileged aspect ObserverProtocol_1
2  {
3     private List observers = new ArrayList();
4     //---------------------------------------
5     protected pointcut newInstance(Screen obs):
6       execution(Screen.new(..)) && target(obs);
7
8     after(Screen obs): newInstance(obs){
9        observers.add(obs);
10    }
11    //---------------------------------------
12    protected pointcut subjectChange() :
13       set(* Line.*)  ||
14       set(* Point.*) ;
15
16    after(): subjectChange() {
17       Iterator it = observers.iterator();
18       while (it.hasNext()){
19          Screen obs = (Screen)it.next();
20          obs.display(
21                thisJoinPoint.getSignature() +
22                                " changed..");
23       }
24    }
25 }
```

B. *Instance-level* Observing

```
1  protected privileged aspect ObserverProtocol_2
2  {
3     private Screen obs;
4
5     public void addObserver(Screen obs) {
6        this.obs = obs;
7     }
8     //---------------------------------------
9     public interface Subject {}
10
11    declare parents: Line      implements  Subject;
12
13    protected pointcut subjectChange(Subject s) :
14       (
15          set(* Line.length)
16       ) && target(s);
17
18    after(Subject s): subjectChange(s) {
19         obs.resize(((Line) s).length);
20    }
21 }
```

has only one observer field (Line 3) that holds a reference copy of the observing instance that will be assigned via the *addObserver* method, which will be invoked at the client application (In particular, at the line(s) where the '$let - observe - exec$' statement is written in the source code). Once the subject has changes in its attributes, the $subjectChange$ pointcut declared in Lines 13-16 is executed. As a result, the observing instance is notified (the advice declared in Lines 18-20 will do this task) using the notification method that was already associated with the statement of the observer construct. In addition, this aspect has a public $Subject$ interface (Line 9) that will be implemented by all observed (Subject) classes. This interface can then used in place of subject classes to capture changes of any subject implementing it. After generating this aspect successfully, the '$let - observe - exec$' statement is replaced by a method-call statement, as follows:

```
ObserverProtocol_2.aspectOf().addObserver(screen1);
```

IV. RESULTS AND DISCUSSION

After implementing our language extension to the examples presented in this paper, we have addressed some of the issues discussed in the literature related to modularity and implementation overhead, and describe how they are handled (at least partially) in our approach. Furthermore, we could identify the characteristics of the proposed observer construct in addition to some future improvements to it.

A. Addressed issues

1) Implementation Overhead: This issue was occurred with several traditional implementations of design patterns (such as, the use of OO or AO constructs) as programmers have to have in mind how the implementation of design patterns should work with their functional code. In our approach, the programmer is not concerned about the concrete implementation of the pattern since it is automatically generated by the extended compiler based on the parameters provided via the pattern construct statements. This lets programmers save time and space and, subsequently, focus on their functional parts of the code (i.e., enhanced productivity). Through the examples illustrated in this paper (and other not reported examples), we strongly believe that our approach will outperform other implementations of the observer pattern proposed in the literature in terms of lines of code (LOC) if applied to larger applications.

Although in Meta-AspectJ [11] programmers can abstract the overall implementation of the observer pattern in AspectJ with fewer lines of code, this would end up with a complex (not expressive) abstraction that imposes users to be aware of the aspects that would be generated. In our approach, programmers are not aware of what is happening inside the aspects. All what they need in our approach is to specify the observing/observed classes or instances along with the desired attributes and notification methods.

2) Modularity: In our approach, modularity is witnessed by separating the implementation of the observer design pattern from the implementation of the actual logic of the application. This means that the actual implementation of the observer pattern is not visible to the programmer and it is also isolated from one application to another. This allows programmers (at different clients) extend, alter and maintain their applications of this design pattern modularly without being aware of what is happening in the background.

It can be observed that our implementation of the observer design pattern satisfies all modularity properties (i.e., locality, reusability, composition transparency, and (un)pluggability) firstly used by Hannemann and Kiczales [3], and thereafter used by Rajan [4], Sousa and Monteiro [12], and Monteiro and Gomes [13]. Our implementation is localized since the overall implementation code of the observer pattern is automatically generated in the compiler background, which means that it is totally separated from the pattern application. It is also reusable because it can be applied to various scenarios without the need to duplicate the source code. Composing a class or an instance in our observer construct will not interfere at all with other classes or instances. Finally, adding (plugging) or removing (unplugging) an application of the observer pattern in a given system using the proposed construct will not require programmers to do changes on other parts of that system.

 ISBN: 978-1-61208-438-1

B. Features

1) Hybrid approach: Our approach combines different features of the approaches proposed in the literature under one roof. It is implemented as a Java/AspectJ extension that summarizes plenty of code in few-keywords constructs, just like meta-AspectJ [11]. Additionally, it automatically generates aspects according to the information provided as parameters in applied construct, adapted from the parametric aspects [14].

2) Expressiveness: The syntax of design patterns is clear, concise and expressive in a way it does not require importing packages, building classes (or aspects), or worrying about something missing in the design principle of the design pattern. All what programmers need to learn in our approach is the construct syntax used for implementing the observer pattern, and also how to apply each scenario using that construct. Furthermore, the readability and writeability is highly improved as the written code becomes shorter and more self-explanatory. So, the absence of dependencies makes it very easy to revise the code for the sake of maintenance.

3) Supporting different levels of application: Supporting different levels of applications of the constructs helps programmers decide where and how to apply constructs. The *class-level* application is beneficial if all instances of a certain class needs to apply the observing functionality, whereas *instance-level* application is useful when certain instances of a class need to apply the observing logic, or when each instance needs to have its own logic.

C. Improvement Considerations in the Future

1) Optimization: As described in the paper, our extended compiler creates a separate aspect for each application of the pattern construct. This potential duplication of generated aspects would require more processing from the compiler, since each aspect node can contain a set of internal AST nodes, which may lead to more compiler passes to be executed. This problem can be resolved in the future by generating a single observer-protocol aspect to handle the implementation of all applications of the observer construct, by employing a particular pointcut and advice for every construct application.

2) Application of the other scenarios: Although the observer construct itself is general enough to capture any observing behavior directly, the current implementation of it does not allow intermixing *class-level* and *instance-level* observing scenarios in a single statement. This means that the implementation requires such scenarios to be specified by more than one '$let - observe - exec$' statement). For example, if observing one subject by multiple observers is needed, then the programer will have to apply the observer construct to each observer with the intended subject (i.e., it will be applied as a set of one-to-one scenarios). Alternatively, the construct should be improved in future to support the other scenarios using single-statement applications.

3) Disabling of pattern application: In our approach, programmers can apply constructs anywhere in their programs. However, to disable a pattern for a certain target, programmers are required to search for the construct application in the program and then comment or remove it. This kind of disabling suffers from a traceability overhead, as the efficient way to this end is to have disable/enable constructs in the future that can automate this action.

V. Related Work

Hannemann and Kiczales [3] used AO constructs to improve the implementations of the original 23 design patterns using AspectJ. They provided an analysis and evaluation of the improvement achieved to the implementation of the patterns according to different metrics, which also have been addressed later by Rajan [4] using Eos extended with the *classpect* construct that unifies class and aspect in one module. When compared with Hannemann's implementation in terms of lines of code and the intent of the design patterns, Rajan observed that Eos could efficiently outperform AspectJ in implementing 7 of the design patterns, while being similar for the other 16 patterns. In addition, the *instance-level* advising feature supported by Eos *classpects* was another advantage over AspectJ. This feature allows a direct representation of runtime instances without the need to imitate their behavior. Another work was also done by Sousa and Monteiro [12] with CaesarJ that supports family polymorphism. Their approach employs a *collaboration* interface that can hold a set of inner abstract classes, and some second level classes: the implementation and binding parts. Also, their results demonstrated positive influence of the *collaboration* interface on modularity, generality, and reusability over those with AspectJ. Gomes and Monteiro [15] and recently in [13] introduced the implementation of 5 design patterns in *Object Teams* compared with that in Java and AspectJ. Regardless of *Object Teams* goals, it showed a powerful support in implementing design patterns efficiently, and with more than one alternative. The entire conversion of aspects into teams was described in detail in their work. The common issue with all these different approaches is that they suffer from the implementation overhead and traceability problems as the concrete implementation of design patterns is required to be manually written by programmers, which may reduce their productivity.

Another approach was introduced by Zook et al. [11]. This approach uses code templates for generating programs as their concrete implementation, called Meta-AspectJ (*MAJ*). Development time is reduced in this approach since it enables expressing solutions with fewer lines of code. With respect to design patterns, *MAJ* provides some general purpose constructs that reduce writing unnecessary code. However, programmers cannot explicitly declare the use of design patterns at certain points of the program, which may also lead to a traceability problem.

Another trend, which is close to our approach, was introduced by Bosch [2], who provided a new object model called *LayOM*. This model supports representing design patterns in an explicit way in C++ with the use of layers. It provides several language constructs that represent the semantics of 8 design patterns and can be extended with other design patterns. Although *LayOM* could resolve the traceability problem and enhance modularity, it lacks expressiveness as it has a complicated syntax consisting of message forwarding processes that might confuse programmers. Our approach seems to provide a similar power to *LayOM*, but, in contrast, the observer construct in our approach has a more concise, expressive, easy-to-use and easy-to-understand syntax.

Hedin [16] also introduced a new technique that is slightly similar to *LayOM* but using rules and pattern roles. The rules and roles can be defined as a class inheritance and specified by attribute declarations. Doing so, it enables the

 ISBN: 978-1-61208-438-1

extended compiler to automatically check the application of patterns against the specified rules. However, the creation of rules, roles, and attributes has a complex syntax that lacks expressiveness and requires an extensive effort to learn and build them.

Another extensible Java compiler is *PEC*, which was proposed by Lovatt et al. [17]. Design patterns in *PEC* were provided as marker interfaces. A class must implement the proper ready-made interface in order to conform to a certain design pattern. After that, the *PEC* compiler will have the ability to check whether the programmer follows the structure and behavior of that pattern or not. However, the *PEC* compiler does not reduce the effort needed to implement design patterns (i.e., it suffers from implementation overhead). Instead, it allows programmers to assign the desired design pattern to a given class and then implement that pattern manually. Eventually, the compiler will just check the eligibility of that implementation.

Budinsky et al. [18] introduced a tool that automates design pattern implementation. Each design pattern has a certain amount of information like name, structure, sample code, when to use, etc. The programmer can supply information about the desired pattern, then its implementation (in C++) will be generated automatically. This approach allows programmers to customize design patterns as needed, but the modularity and reusability is missed, and it suffers from the traceability problem as well.

VI. Conclusion

This paper introduces two contributions in regards to the observer design pattern. Firstly, it presents a detailed classification of all possible scenarios of the observer pattern that might be utilized in various kinds of applications. Secondly, a new approach for implementing the observer pattern in Java is proposed to cover a partial set of the scenarios introduced. This approach is developed as a language extension (Java/AspectJ extension) using *abc*. The syntax, semantics and application of the proposed observer construct are illustrate in detail, and by means of typical examples, we demonstrate how the implementation of the observer pattern using this approach has been simplified and has become conciser, more expressive and more modular. The capabilities and advantages of the proposed approach seem promising and we anticipate that our approach will supersede current approaches.

We hope in the future to improve this approach by following the recommendations provided in the paper. This includes supporting the application of other scenarios of the observer pattern using a single statement rather than many. Resolving current issues of the observer pattern implementation such as optimization and application disabling will also be taken into account in the future. Furthermore, an evaluation of our proposed approach compared to with other approaches proposed in the literature is another important objective that we hope to achieve in future. Moreover, we aim to develop constructs for implementing other software design patterns to make their implementations more expressive and modular.

References

[1] J. Vlissides, R. Helm, R. Johnson, and E. Gamma, "Design patterns: Elements of reusable object-oriented software," Reading: Addison-Wesley, vol. 49, 1995, p. 120.

[2] J. Bosch, "Design patterns as language constructs," Journal of Object-Oriented Programming, vol. 11, no. 2, 1998, pp. 18–32.

[3] J. Hannemann and G. Kiczales, "Design pattern implementation in Java and AspectJ," in ACM Sigplan Notices, vol. 37. ACM, 2002, pp. 161–173.

[4] H. Rajan, "Design pattern implementations in Eos," in Proceedings of the 14th Conference on Pattern Languages of Programs. ACM, 2007, pp. 9:1–9:11.

[5] P. Avgustinov et al., "abc: An extensible aspectj compiler," in Transactions on Aspect-Oriented Software Development I. Springer, 2006, pp. 293–334.

[6] N. Nystrom, M. R. Clarkson, and A. C. Myers, "Polyglot: An extensible compiler framework for Java," in Compiler Construction. Springer, 2003, pp. 138–152.

[7] A. Mehmood and D. N. Jawawi, "Aspect-oriented model-driven code generation: A systematic mapping study," Information and Software Technology, vol. 55, no. 2, 2013, pp. 395–411.

[8] N. Cacho et al., "Blending design patterns with aspects: A quantitative study," Journal of Systems and Software, vol. 98, 2014, pp. 117–139.

[9] R. Vallée-Rai et al., "Optimizing java bytecode using the soot framework: Is it feasible?" in Compiler Construction. Springer, 2000, pp. 18–34.

[10] "Java 1.2 parser for CUP." [Online]. Available: https://github.com/Sable/abc/blob/master/aop/abc/src/abc/aspectj/parse/java12.cup Last access: September 15, 2015.

[11] D. Zook, S. S. Huang, and Y. Smaragdakis, "Generating AspectJ programs with meta-AspectJ," in Generative Programming and Component Engineering. Springer, 2004, pp. 1–18.

[12] E. Sousa and M. P. Monteiro, "Implementing design patterns in CaesarJ: an exploratory study," in Proceedings of the 2008 AOSD workshop on Software engineering properties of languages and aspect technologies. ACM, 2008, pp. 6:1–6:6.

[13] M. P. Monteiro and J. Gomes, "Implementing design patterns in Object Teams," Software: Practice and Experience, vol. 43, no. 12, 2013, pp. 1519–1551.

[14] K. Aljasser and P. Schachte, "ParaAJ: toward reusable and maintainable aspect oriented programs," in Proceedings of the Thirty-Second Australasian Conference on Computer Science-Volume 91. Australian Computer Society, Inc., 2009, pp. 65–74.

[15] J. L. Gomes and M. P. Monteiro, "Design pattern implementation in Object Teams," in Proceedings of the 2010 ACM Symposium on Applied Computing. ACM, 2010, pp. 2119–2120.

[16] G. Hedin, "Language support for design patterns using attribute extension," in Object-Oriented Technologys. Springer, 1998, pp. 137–140.

[17] H. C. Lovatt, A. M. Sloane, and D. R. Verity, "A pattern enforcing compiler (PEC) for Java: using the compiler," in Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling-Volume 43. Australian Computer Society, Inc., 2005, pp. 69–78.

[18] F. J. Budinsky, M. A. Finnie, J. M. Vlissides, and P. S. Yu, "Automatic code generation from design patterns," IBM Systems Journal, vol. 35, no. 2, 1996, pp. 151–171.

 ISBN: 978-1-61208-438-1

Supporting Tools for Managing Software Product Lines: a Systematic Mapping

Karen D. R. Pacini
and Rosana T. V. Braga

Institute of Mathematics and Computer Sciences
University of São Paulo
São Carlos, SP, Brazil
Email: karenr@icmc.usp.br, rtvb@icmc.usp.br

***Abstract*—In order to successfully design and build a Software Product Line (SPL), besides the difficult task of making a good domain engineering based on a solid knowledge – both theoretical and practical – about the subject domain, it is still necessary to consider other barriers such as lack of computational support, lack of documentation available and the complexity or unavailability of existing supporting tools. These are some of the reasons that may discourage the adoption and wide usage of SPL in organizations. In this context, this paper presents a Systematic Mapping (SM) of supporting tools for managing SPLs. This SM was performed in order to identify, gather and classify existing solutions in the literature that offer support for managing product lines both in single or multiple phases, since conception until product derivation and evolution of the SPL. The information gathered about the solutions selected is presented in the results. This information comprises the completeness of the solutions, their complexity and quality, and also points out their benefits and limitations. It is expected as the result of this SM an overview of SPL management solutions in order to support developers and SPL engineers to find suitable options to apply in their projects, in addition to highlight gaps on the research area and suggest future works.**

Keywords*—*Software Product Line Management; Systematic Mapping.

I. INTRODUCTION

The software industry has been adapting to the large increase of demand arising from the constant evolution of technology. The concept of software reuse gets an important role on this new way of software manufacturing, in which development time is reduced, while quality is improved [1]. Over time, many approaches have emerged trying to achieve this goal, such as: object-oriented paradigm, component-based development, service-oriented architecture, among others.

Software product lines (SPL) emerged in this context, to support reuse by building systems tailored specifically for the needs of particular customers or groups of customers. Reuse in SPL is systematic, it is planned and executed for each artifact resulting from the development process. According to Ezran et al. [2] these artifacts encapsulate knowledge and are very valuable to the organizations, because they are an interrelated collection of software products that can be reused across applications.

The most common SPL development approaches, such as Product Line UML-Based Software Engineering (PLUS) [3], Product Line Practice (PLP) [1], etc., are focused on the process to support the domain engineering and/or the application engineering, without considering the computational tools that support the process. Thus, the choice and use of tools are apart from the process and strongly associated to phases of definition of the feature models and its mapping to the artifacts that implement them. Some examples of these tools include Pure::Variants [4], Gears [5], and GenArch [6].

Supporting tools offer the developers a complete environment for development and maintenance of the product line, aiming at facilitating its adoption. When supporting tools are employed for SPL management, from conception to evolution, developers can dedicate more attention to the development itself, i.e., to domain and application engineering. This can help improving production quality for both the product line and its generated products, deviating the focus from the development environment or other more specific management questions. Although there is a huge variety of existing tools, it is not possible to ensure that the needs of SPL engineers are being fulfilled. It is necessary to better investigate the scope, the availability and the specificity of these supporting tools in order to identify gaps to be filled.

Therefore, this paper presents a Systematic Mapping (SM) of supporting tools for managing SPLs in order to identify and analyse the solutions that exist in the literature. The identified solutions can support the management of SPL from conception to development, maintenance and evolution phases. The analysis proposed in this paper intends to offer an updated overview of the existing supporting tools and identify perspectives of researches related to product lines.

Section II presents some relevant related works that resemble this work in terms of searching for tools that support somehow the development of SPL. Section III presents the process of the SM, with the corresponding phases: planning, conducting and reporting. Section IV presents the summarization and the data analysis observed from the studies selected from the SM. Section V presents a discussion about the analysis of the gathered solutions, pointing out its benefits and limitations. Finally, Section VI presents the conclusions of this research.

II. RELATED WORKS

SPLs have become a popular concept nowadays. Despite the difficulty on its adoption, there are big investments and research improvements in this area. Therefore, there are many

 ISBN: 978-1-61208-438-1

attempts to catalogue the whole resources available, as surveys, SMs and systematic reviews. Each performed research has a set of requirements for the search that defines the focus and the granularity of the target tools. In this section, we present three related works that resemble this work, but with some differences in their purpose and nature of the search.

Lisboa et al. [7] performed a systematic review on supporting tools for domain analysis only. The findings were evaluated according to the type of the offered support, its completeness and quality, as well as if it fulfills all that is expected. This work followed the process proposed by Kitchenham [8] to perform a systematic review of the literature, with 19 tools selected. The evaluation of these tools considered the provided functionality, documentation, interface, user guide, among others. As the result, this review provided to the user a guideline to find the supporting tool that suits expectations regarding domain analysis, however it is limited to this phase.

Munir et al. [9] performed a survey to identify and evaluate supporting tools for development and maintenance of SPLs according to a predefined set of requirements. The study resulted in 13 tools, but only two were chosen for evaluation considering quality factors. As the result, some gaps were highlighted on this area, besides a set of available tools with free or commercial licenses. However, this work is a little outdated (2010) and does not provide a comparison between all selected tools, showing only a brief description of each one.

Djebbi et al. [10], on the other hand, performed an industry survey in order to find supporting tools for the management of product lines, very similar to the proposal of this paper. The survey considered requirements, quality and open issues for a case study and evaluation. However, in this survey, they considered only relevant tools in the industry context, thus, only four tools were selected. For each tool evaluation, preselected criteria were applied in case studies using the tool. As a result, the evaluation of each tool is presented, which highlights its priorities and application context guiding the stakeholders' choice for the most suitable tool regarding their necessity.

Therefore, this paper proposes a research of the state of the art regarding the solutions that support the management of SPLs, considering the type of support provided, the coverage of phases, the specificity, among others. Solutions that contribute to the management, even if they are still in design, were considered and no constraint regarding the context has been established. As the result, statistics of the findings are provided.

III. Systematic Mapping

SM, according to Kitchenham et al., is a broad review of primary studies in a specific topic area that aims to identify what evidence is available on the topic [11]. It is very similar to another process proposed by same author called Systematic Literature Review (SLR) [8]. Although mapping studies use the same basic methodology as SLRs, its main purpose is to identify all research related to a specific topic rather than addressing the specific questions that SLRs address [12]. Thus, mapping studies can be of great potential importance to software engineering researchers by providing an overview of the literature in specific topic areas [13], where the quantity of the selected studies, their type, the available results, the frequency of publications through time, among others, can be observed [14].

This SM followed the same process proposed by Kitchenham for SLRs [8], which contains three main phases: 1-Planning: the research objectives and the SM protocol are defined; 2-Conducting: the primary studies are identified, selected, and evaluated according to the inclusion and exclusion criteria previously established; and 3-Reporting: a final report is organized and presented according to the data extraction and analysis.

A tool for supporting the SM process called StArt (State of the Art through Systematic Review) [15] was used to manage the whole execution of this SM. Figure 1 illustrates how StArt deals with the several SM phases. The full protocol and SM StArt File with the full process, all outcomes (including the list of the 1046 works, where the 50 selected ones appear in the respective Extraction phase), filled forms and other details can be found elsewhere for further reference [16]. This extra material allows the reproduction of the study whenever necessary, e.g., to update it or to evolve it to a SLR.

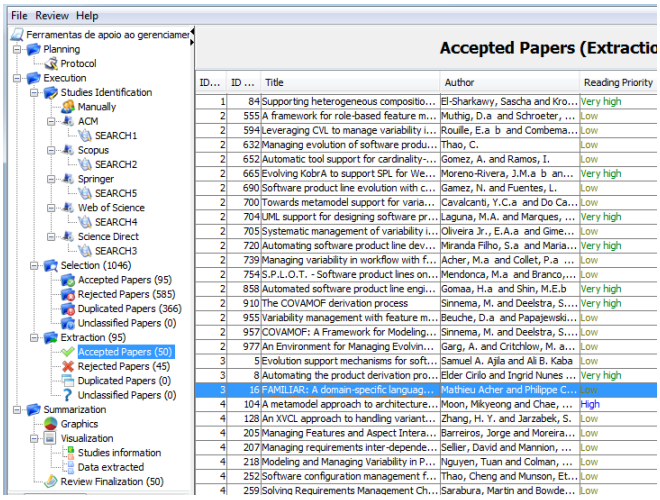


Figure 1. Using StArt to support the SM.

The next subsections present the three main phases of the SM process in detail.

A. Planning

In this phase, the SM protocol is established, which consists of: research objectives; research questions, range, and specificity; sources selection criteria; studies definition; and procedures for studies selection.

1) Objective: This SM aims to investigate the state of the art with respect to computational support to the SPL management, regarding the phases of development, maintenance and evolution of SPL, in order to identify the quantity and the quality of the solutions currently available, considering its completeness and complexity, as well as highlighting its benefits and limitations.

2) Research Question: Aiming at finding possibly all primary studies to understand and summarize evidences about existing solutions for SPLs management, the following research question (RQ) was established:

RQ1: What are the existing solutions in the literature that present a computational support to the management of SPL?

a) Does the presented solution use patterns or known standards?
b) What is the technology used for SPL management presented by the solution?
c) Which phases of the SPL life cycle are supported by the solution?

By management we mean inclusion, modification, removal, or search of all the artifacts produced during the SPL engineering (domain engineering or application engineering), as well as its evolution after the SPL is delivered.

3) Inclusion and Exclusion Criteria: The Inclusion Criteria (IC) and the Exclusion Criteria (EC) make it possible to include primary studies that are relevant to answer the research questions and exclude studies that do not answer them. Thus, the inclusion criteria of this SM are:

- **IC1:** Studies that present a tool, approach, technique, process, method or any other software engineering resource that offers a solution for management of one or more phases of the SPL life cycle.
- **IC2:** Studies that present a proposal of a solution for managing one or more phases of the SPL life cycle, even if it is yet in the project phase.

Criteria IC2 reinforces that any practical solution that could effectively help the SPL management, even if the solution is still in project, is included, as it could give us insights about important issues to consider. The term practical, in this context, does not mean a concrete solution, but purely a solution that directly helps the SPL management.

Non-relevant studies with respect to the objectives of this SM are discarded applying one of these four defined exclusion criteria:

- **EC1:** Studies that do not present a solution, consolidated or not, for the management of one or more phases of the SPL life cycle.
- **EC2:** Studies that are short versions of a published full work.
- **EC3:** Studies that are incomplete, unavailable and/or duplicated (multiple instances of same document).
- **EC4:** Studies that describe events, or are an event index or schedule.

For studies classification, i.e., inclusion or exclusion, it is mandatory to apply one of the defined criteria above. However, if more then one criteria is applicable to a particular study, this formula is used: **((IC1) OR (IC2)) AND NOT ((EC1) OR (EC2) OR (EC3) OR (EC4))**.

4) Sources Selection Criteria Definition: The sources choice was made considering their relevance in the software engineering area and also a specialist opinion in cases of conferences, books and workshops that are not indexed in the search engines. The specialist opinion included in several activities relied on the participation of the co-author of this paper. For the search engines, we considered those with an updated content, with availability of the studies, with an advanced search mechanism, with quality results and with flexibility to export the findings. Thus, the following search engines were selected: *IEEE Xplore*, *ACM Digital Library*, *Science Direct*, *Scopus* and *Web of Science*.

5) Search String Construction: From a group of studies selected by the specialist, called in the SM process the 'control group', in addition to the objective of this SM, the search string was defined according to Table I and resulted in: **(A) AND (B) AND (C) AND (D)**.

TABLE I. TERMS, KEYWORDS AND SYNONYMS.

Term	Keyword	Synonym
A	Product Line	"SPL", "Product Lines", "Product Family", "Product Families", "Product-Line", "Product-Lines", "Product-Family", "Product-Families"
B	Tool	"tools", "tool-supported", "support", "supported", "supporting"
C	Management	"manage", "managing", "storage", "repository"
D	Software	-

B. Conducting

This phase was performed right after the protocol definition and was divided into two phases of selection. It was carried out between November, 2013 and January, 2014. The first phase is called studies identification, and it defines the group of studies that will be used as base to the second phase, which is called studies selection. In the first phase of our SM, the search was performed using five search engines (ACM Digital Library, IEEE Xplore, Science Direct, Scopus, and Web of Science), according to the previously defined string. The specific syntax was used in each search base considering only the title, the keywords and the abstract. This resulted in 1046 works.

In the second phase, we applied the inclusion and exclusion criteria defined in the SM research protocol. After the selection is performed, a refined group of studies is obtained according to the context of the SM for the data extraction.

The selection phase of this SM was divided into three parts. In the first selection the inclusion and the exclusion criteria were applied to the base group of identified studies considering only title and abstract, which resulted in the inclusion of 95 studies and the exclusion of 951 studies. Among the excluded studies, 366 were marked as duplicated and 585 were excluded by other exclusion criteria, resulting in 95 studies. In the second selection, the full text was considered, and 41 studies were included while 54 were excluded. The third selection used the specialist opinion in order to validate the selected studies and therefore, 9 previously excluded studies were included again by consensus. The Figure 2 and the Figure 3 present the total application of inclusion and exclusion criteria, respectively, in the selection of studies by search engine.

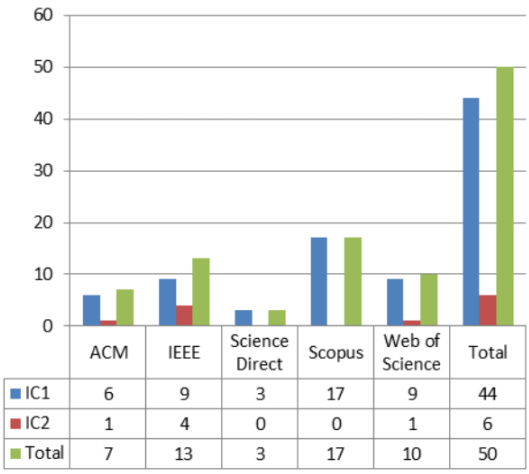
After the selection of the studies included in the SM, it is possible to proceed to the next phase of the process, where data extraction is performed.

C. Reporting

This phase aims to extract and analyse the data in order to organize and present a final report about the findings. The data extraction summarizes the data of the selected studies for further analysis. For the data extraction performed in this SM, an extraction form filled after reading each paper was used.

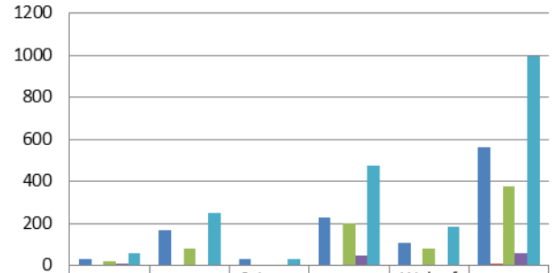
This form is intended to help answering the research questions of this SM. In addition to collecting the basic information about the studies, such as title, author, year and type of publication (journal, conference, etc.,), the form also

 ISBN: 978-1-61208-438-1



	ACM	IEEE	Science Direct	Scopus	Web of Science	Total
IC1	6	9	3	17	9	44
IC2	1	4	0	0	1	6
Total	7	13	3	17	10	50

Figure 2. Included studies by search base x inclusion criteria



	ACM	IEEE	Science Direct	Scopus	Web of Science	Total
EC1	31	168	29	225	108	561
EC2	0	2	0	4	0	6
EC3	19	77	3	198	77	374
EC4	7	0	0	48	0	55
Total	57	247	32	475	185	996

Figure 3. Excluded studies by search base x exclusion criteria

collects specific information useful for this research. Among the specific information, we can mention the type of solution presented by the study, a brief description of the solution, with its benefits and limitations, its specificity, the handled phases, the software engineering resources used in the solution, the use of patterns/standards, validation and managed parts.

IV. Result Analysis

After a careful data extraction, sufficient information was gathered to perform an analysis of the results. The analysis of the selected studies was divided according to the characteristics described in the extraction form.

A. Full Analysis of the Studies

The analysis was divided into two parts, the first considers all solutions observed in the phase of data extraction, while the second part performs a more detailed review only on the most complete solutions.

1) Frequency of Publication: It was observed that the frequency of publications of solutions for the management of SPLs has increased significantly from 2008 to its peak in 2011, and then remained stable. This trend can be graphically observed in Table II, indicating that this is an area of current research interest.

Table II. Summarization of studies divided by year of publication.

2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
1	0	1	3	2	1	3	2	9	5	6	9	5	3	50

2) Specificity: A great part of the solutions found are specific to support only a particular cycle of SPL management, as can be observed in Table III. Among the specificity, we can highlight the variability management. In fact, most solutions available both commercially and freely are geared to support variability management, providing processes, models, methods, approaches, tools, among others. Examples are KobrA [17] and COVAMOF [18].

Table III. Summarization of studies divided by specificity of the solution.

Architecture	Asset	Compatibility	Configuration/ Derivation
1	1	1	4
Evolution	**Extraction**	**Feature**	**MultipleSPL**
5	1	7	2
Requirement	**Variability**	**Versioning**	**None**
1	18	1	10

3) Patterns/standard Use: Among the observed solutions the lack of pattern/standards used during the development becomes evident, as shown in Table IV. This may hinder the flexibility of the solution, and even its adoption and integration with other existing solutions. Although the use of patterns or standards may generate an additional effort, their use to develop a solution is strongly recommended, as the effort would be rewarded in the future with enhanced system maintenance and evolution.

Table IV. Summarization of studies divided by usage of patterns or standards.

RAS	OCL	CVL	ADL	CVA
1	2	3	1	1
MOF	**OVM**	**VML**	**QVT**	**None**
2	1	1	2	40

The standards identified in the solutions presented in Table IV are: Reusable Asset Specification (RAS) [19], Object Constraint Language (OCL) [20], Common Variability Language (CVL) [21], Architecture description language (ADL) [22], Commonality and Variability Analysis (CVA) [23], Meta-Object Facility (MOF) [24], Orthogonal Variability Model (OVM) [25], Variability Language VML [26] and Query/View/Transformation (QVT) [27].

4) Phases Supported: From the 50 studies selected, only 17 offer support to both development phases (domain engineering and application engineering), in the remainder, five support only application engineering and 28 only domain engineering.

The information extracted from these 17 more relevant studies, which support both phases of development, are presented in detail in Table V, and are evaluated more specifically in the second phase of the analysis.

 ISBN: 978-1-61208-438-1

B. Selected Studies Analysis

Among the 17 solutions presented by the studies, 15 offer support to the development of the SPL, while two offer support only to maintenance and evolution of existing SPL. From the 15 solutions that support the development phase, one supports only this process and 14 offer support to maintenance too. Among these 14 solutions, three support only these two activities, one supports also the evolution of the SPL, one supports also the derivation of products and nine additionally offer a visualization of the SPL with its variability. Among these nine, only one also offers support for evolution.

Only two solutions offer support from the development to the maintenance and evolution of product lines, and only one of them offers an overview of the variability and the produced assets.

Thereby, the lack of solutions that support all phases of development and evolution of SPL is evident, which highlights a gap to be investigated.

V. Discussion

The existing solutions in the literature for computational support to SPLs were identified in the SM performed in this work. In addition, it was possible to identify and analyse the major benefits and limitations of the overall solutions selected.

Among the benefits observed, we highlight that most of the solutions intend to provide guidance to resolve the problem or part of it, besides providing an execution flow to achieve a goal. These flows are presented as a process or even wizards, which allows users, both beginners and experts, to operate in a particular management area, such as SPL development or evolution, ensuring that the process was correctly executed and that its results are consistent, since the provided management is explicit and organized.

Besides that, seven solutions support the user when making decisions and understanding the process. Regarding the decision making support, six solutions provide the users both overviews and specific views from the current state of the SPL, highlighting the chosen objects to be observed.

Maintaining the consistency is also an issue treated by most solutions found (13 solutions), some establish constraints, inspections and validations, some even provide the tracking of the assets, which supports also the SPL evolution.

It is also worth to mention the great reduction of costs and effort provided by the nine solutions that offer various automated functionality to support the management. Five solutions even offer resources to significantly reduce the complexity of tasks that would be very laborious to perform manually.

The leverage of reuse is also an issue treated by 12 solutions. These solutions often provide configuration, importation and exportation options, compatibility with others tools and sharing of assets among product lines, which promotes the interoperability and leverage the reuse, both inside the solution itself and out of it.

However, despite the many benefits observed, it was possible to identify many limitations that often can discourage the use of the solution. Among the main limitations, the lack of complete solutions to manage SPLs is highlighted. Most of the analysed solutions (16 solutions) are focused on the resolution of a single issue, that is, provide support only to manage a specific area. Besides that, six solutions have a very marked limitation regarding the scope to address an area and can be very specific, for example, manage only requirements, and furthermore do not offer the possibility of extension or parametrization to generalize their use.

The authors of seven solutions declare that their use may be more complex than desired because, to use the solution, the user needs to study and master its specifications, which very often discourage its adoption. Besides the complexity, a barrier to be also considered is that most solutions analysed (10 solutions) need a great intervention of a specialist that understands the models and specifications and performs many manual procedures.

One of the limitations strongly considered as motivation for the adoption of a solution is the lack of a graphical interface, or an interface that is very complex or poor, as occurs in four solutions. One of the solutions does not even support version control, which makes the SPL maintenance and evolution very difficult.

A great barrier, mostly at the academy, is the fact that various solutions are private [37][40]. Unfortunately, those are the solutions that provide the most complete group of solutions for SPL management, as well as user support and validation. Some freely available solutions analysed have not been even validated or implemented yet.

VI. Conclusion and Future Works

This work aimed to identify and evaluate existing solutions in the literature to support the management of SPLs. For this, a SM was performed, in which the research protocol was defined and followed for conducting the research. From the data extracted from the selected studies, it was possible to gather enough information to answer the research questions proposed in this paper.

The outcomes of the research present 50 existing solutions in the literature that provide computational support for at least one phase of SPL management, which represents a very reduced offer of solutions to support the management of SPL. Most of the existing solutions do not offer a complete support, not covering the whole phases of the development, maintenance and evolution of the SPL and even not providing a great usage of the whole reuse potential that artifacts and features may offer.

The contribution of this paper is interesting both to the academic and to the professional segment. In the academic environment, this research helps to highlight the lack of complete solutions in this area, in addition to highlight the lack of standards on these existing solutions and the lack of validation, which hinders its use. In the professional context, this research helps professionals to find potential tools that will help them deploy a product line or even help them manage existing product lines.

The gaps pointed out in this SM suggest perspectives of future works, of which we can highlight:

- *Leverage of reuse:* Among all the presented solutions, only one [30] allows the sharing of the assets in the repository between SPLs. An approach where the stored assets may not be bounded into a specific SPL and could be available to be freely used in any context

 ISBN: 978-1-61208-438-1

TABLE V. DATA ANALYSIS OF THE MOST RELEVANT STUDIES

Title	Year	Search Base	Type of Solution	Variability Management	Other Specificity	Uses Pattern/Standard	Validated	Development Support	Maintenance Support	Evolution Support
An approach to variability management ... [28]	2012	Scopus	Tool	•				•	•	
Automated software product line engineering ... [29]	2007	IEEE	Tool				•	•	•	
Automating software product line development... [30]	2010	IEEE	Tool			•	•	•	•	
Automating the product derivation process ... [31]	2012	ACM	Approach	•	•	•	•	•	•	
Evolving KobrA to support SPL for WebGIS development [17]	2011	IEEE	Tool	•		•	•	•	•	
Feature model to product architectures ... [32]	2009	Scopus	Process				•	•	•	•
FLiP: Managing Software Product Line Extraction ... [33]	2008	Scopus	Tool		•				•	
Holmes: a system to support software product lines [34]	2000	Scopus	Tool					•	•	
Involving Non-Technicians in Product Derivation ... [35]	2007	Web of Science	Tool	•	•			•	•	
Modeling and Building Software Product Lines with Pure ... [36]	2008	IEEE	Tool				•	•	•	
New methods in software product line development [37]	2006	Web of Science	Method					•	•	
Support for complex product line populations [38]	2011	Scopus	Tool		•			•	•	
Supporting heterogeneous compositional multi software ... [39]	2011	Scopus	Tool				•	•	•	
The BigLever Software Gears Unified Software Product ... [40]	2008	Web of Science	Framework				•	•	•	
The COVAMOF derivation process [18]	2006	IEEE	Framework	•	•		•		•	
Toward an Architecture-Based Method for Selecting ... [41]	2010	Web of Science	Framework		•		•	•		
UML support for designing software product lines ... [42]	2010	IEEE	Tool			•	•	•	•	•

would leverage the reuse not just inside the SPL itself but widely.

- *Use of patterns/standards:* The lack of use of standards was evidenced in this research. A solution designed taking into account standards or patterns could provide more safety and organization for the user, besides promoting the flexibility and interoperability with other tools. As an example of the use of standards in solutions focused on SPLs, we highlight RAS [19] and CVL [21].
- *Use of services:* The service-oriented usage is also a good option in order to promote the interoperability among tools and facilitate the access to the functionality of the solution. This is a resource that was not found in most solutions analysed, except for the solution proposed by S. Khoshnevis [28].
- *A free complete solution to support SPL management:* The creation and provision of a free and complete approach to support the management of SPLs, with validation, documentation and adequate support could rise the community interest in adoption of the SPL concept, both for developing and usage.
- *Possibility of extending the SM towards a complete SLR:* The SM main goal was to identify all research related to a specific topic. Based on the resulting complete protocol that is publicly available [16], a complete SLR can be performed to address specific questions.

The solutions that can be built as a result of this SM could incentive the adoption of SPL in both academical and professional environments, in addition to leveraging the reuse of assets in contexts other than SPL. As the next step of this research, it is planned to analyse existing solutions, by running performance and quality tests in order to compare and evaluate them through a full systematic review about supporting tools for the management of SPLs. Then, if none of the solutions are considered adequate, we aim to propose a new solution that fulfills developers needs.

ACKNOWLEDGMENTS

Our thanks to Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and University of São Paulo (USP) for financial support.

REFERENCES

[1] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns. Addison Wesley Professional, 2002, the SEI series in software engineering.

[2] M. Ezran, M. Morisio, and C. Tully, Practical software reuse. Springer, 2002.

[3] H. Gomaa, "Designing software product lines with uml 2.0: From use cases to pattern-based software architectures," in Reuse of Off-the-Shelf Components. Springer, 2006, pp. 440–440.

[4] D. Beuche, "Modeling and building software product lines with pure:: variants," in 16th International Software Product Line Conference-Volume 2. ACM, 2012, pp. 255–255.

[5] R. Flores, C. Krueger, and P. Clements, "Mega-scale product line engineering at general motors," in 16th International Software Product Line Conference-Volume 1. ACM, 2012, pp. 259–268.

[6] E. Cirilo, U. Kulesza, and C. J. P. de Lucena, "A product derivation tool based on model-driven techniques and annotations." J. UCS, vol. 14, no. 8, 2008, pp. 1344–1367.

[7] L. B. Lisboa, V. C. Garcia, D. Lucrédio, E. S. de Almeida, S. R. de Lemos Meira, and R. P. de Mattos Fortes, "A systematic review of domain analysis tools," Information and Software Technology, vol. 52, no. 1, 2010, pp. 1–13.

[8] B. Kitchenham, "Procedures for performing systematic reviews," pp. 1–28, 2004.

[9] Q. Munir and M. Shahid, "Software product line: Survey of tools," Master's thesis, Linköping University, Department of Computer and Information Science, 2010.

[10] O. Djebbi, C. Salinesi, and G. Fanmuy, "Industry survey of product lines management tools: Requirements, qualities and open issues," in 15th IEEE International Requirements Engineering Conference (RE). IEEE, 2007, pp. 301–306.

 ISBN: 978-1-61208-438-1

[11] S. Keele, “Guidelines for performing systematic literature reviews in software engineering,” in Technical report, Ver. 2.3 EBSE Technical Report. EBSE, 2007.

[12] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, “Using mapping studies in software engineering,” in 20th Annual Workshop on Psychology of Programming Interest Group (PPIG), vol. 8, 2008, pp. 195–204.

[13] B. A. Kitchenham, D. Budgen, and O. P. Brereton, “The value of mapping studies: a participantobserver case study,” in 14th international conference on Evaluation and Assessment in Software Engineering. British Computer Society, 2010, pp. 25–33.

[14] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” in 12th International Conference on Evaluation and Assessment in Software Engineering, vol. 17, 2008, p. 1.

[15] A. Zamboni, A. Thommazo, E. Hernandes, and S. Fabbri, “Start - uma ferramenta computacional de apoio à revisão sistemática,” Proc.: Congresso Brasileiro de Software (CBSoft’10), Salvador, Brazil, 2010, pp. 91–96, [English title] StArt - A Computational Supporting Tool for Systematic Review. [Online]. Available: http://homes.dcc.ufba.br/ flach/docs/Ferramentas-CBSoft-2010.pdf

[16] K. D. R. Pacini and R. T. V. Braga. Protocol of the systematic mapping on supporting tools for managing software product lines. [Online]. Available: https://goo.gl/RvAh2T [Retrieved: Sep, 2015]

[17] J. b. Moreno-Rivera, E. Navarro, and C. Cuesta, “Evolving KobrA to support SPL for WebGIS development,” LNCS (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7046 LNCS, 2011, pp. 622–631.

[18] M. Sinnema, S. Deelstra, and P. Hoekstra, “The COVAMOF derivation process,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 4039 LNCS, 2006, pp. 101–114.

[19] O. M. Group, “Reusable asset specification,” OMG, 2005. [Online]. Available: http://www.omg.org/spec/RAS/2.2 [Retrieved: Sep, 2015]

[20] ——, “Object constraint language,” OMG, 2014. [Online]. Available: http://www.omg.org/spec/OCL/2.4 [Retrieved: Sep, 2015]

[21] ——, “Common variability language,” OMG, 2012. [Online]. Available: http://www.omgwiki.org/variability/lib/exe/fetch.php?id=start&cache=cache&media=cvl-revised-submission.pdf [Retrieved: Sep, 2015]

[22] M. Leclercq, A. E. Ozcan, V. Quema, and J.-B. Stefani, “Supporting heterogeneous architecture descriptions in an extensible toolset,” in 29th International Conference on Software Engineering (ICSE). IEEE, 2007, pp. 209–219.

[23] D. M. Weiss and C. T. R. Lai, Software Product-line Engineering: A Family-based Software Development Process. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[24] O. M. Group, “Meta object facility,” OMG, 2014. [Online]. Available: http://www.omg.org/spec/MOF/2.4.2 [Retrieved: Sep, 2015]

[25] F. Van Der Linden and K. Pohl, “Software product line engineering: Foundations, principles, and techniques,” 2005.

[26] N. Loughran, P. Sánchez, A. Garcia, and L. Fuentes, “Language support for managing variability in architectural models,” in Software Composition. Springer, 2008, pp. 36–51.

[27] O. M. Group, “Query/view/transformation,” OMG, 2014. [Online]. Available: http://www.omg.org/spec/QVT/1.2 [Retrieved: Sep, 2015]

[28] S. Khoshnevis, “An approach to variability management in service-oriented product lines,” in 34th International Conference on Software Engineering (ICSE), 2012, pp. 1483–1486.

[29] H. Gomaa and M. Shin, “Automated software product line engineering and product derivation,” 2007. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-39749091282&partnerID=40&md5=34bd6a13ca7198265f69a098d6a0a7e0

[30] S. Miranda Filho, H. Mariano, U. Kulesza, and T. Batista, “Automating software product line development: A repository-based approach,” 2010, pp. 141–144. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-78449273654&partnerID=40&md5=d23375c54254dd7a8a1f1c92b42c7932

[31] E. Cirilo, I. Nunes, U. Kulesza, and C. Lucena, “Automating the product derivation process of multi-agent systems product lines,” Journal of Systems and Software, vol. 85, no. 2, 2012, pp. 258 – 276. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121211001075

[32] D. Perovich, P. Rossel, and M. Bastarrica, “Feature model to product architectures: Applying MDE to software product lines,” in Joint Working IEEE/IFIP Conference on Software Architecture/European Conference on Software Architecture WICSA/ECSA, 2009, pp. 201–210.

[33] V. Alves, F. Calheiros, V. Nepomuceno, A. Menezes, S. Soares, and P. Borba, “FLiP: managing software product line extraction and reaction with aspects,” in 12th International Software Product Line Conference (SPLC), 2008, pp. 354–354.

[34] G. Succi, J. Yip, E. Liu, and W. Pedrycz, “Holmes: A system to support software product lines,” in 22nd International Conference on Software Engineering. New York, NY, USA: ACM, 2000, pp. 786–786. [Online]. Available: http://doi.acm.org/10.1145/337180.337641

[35] R. Rabiser, D. Dhungana, P. Grunbacher, K. Lehner, and C. Federspiel, “Involving non-technicians in product derivation and requirements engineering: A tool suite for product line engineering,” in 15th IEEE International Requirements Engineering Conference (RE), 2007, pp. 367–368.

[36] D. Beuche, “Modeling and building software product lines with pure::variants,” in 12th International Software Product Line Conference (SPLC), Sept 2008, pp. 358–358.

[37] C. Krueger, “New methods in software product line development,” in 10th International Software Product Line Conference (SPLC), 2006, pp. 95–99.

[38] S. El-Sharkawy, C. Kröher, and K. Schmid, “Support for complex product line populations,” in 15th International Software Product Line Conference, Volume 2, ser. SPLC ’11. New York, NY, USA: ACM, 2011, p. 47:1–47:1. [Online]. Available: http://doi.acm.org/10.1145/2019136.2019191 [Retrieved: Sep, 2015]

[39] ——, “Supporting heterogeneous compositional multi software product lines,” in 15th International Software Product Line Conference, Volume 2, ser. SPLC ’11. New York, NY, USA: ACM, 2011, p. 25:1–25:4. [Online]. Available: http://doi.acm.org/10.1145/2019136.2019164 [Retrieved: Sep, 2015]

[40] C. Krueger, “The BigLever software gears unified software product line engineering framework,” in 12th International Software Product Line Conference (SPLC), 2008, pp. 353–353.

[41] M. Tanhaei, S. Moaven, and J. Habibi, “Toward an architecture-based method for selecting composer components to make software product line,” in 7th International Conference on Information Technology: New Generations (ITNG), 2010, pp. 1233–1236.

[42] M. Laguna and J. Marqués, “Uml support for designing software product lines: The package merge mechanism,” Journal of Universal Computer Science, vol. 16, no. 17, 2010, pp. 2313–2332.

 ISBN: 978-1-61208-438-1

Recovering Lost Software Design with the Help of Aspect-based Abstractions

Kiev Gama
Centro de Informática
Universidade Federal de Pernambuco (UFPE)
Recife, PE
email: kiev@cin.ufpe.br

Didier Donsez
Laboratoire d'Informatique de Grenoble
University of Grenoble
Grenoble, France
email: didier.donsez@imag.fr

***Abstract*— In this paper, we propose an unconventional usage of aspect-oriented programming, presenting and discussing a novel approach for recovering layered software design. It consists of a reengineering pattern based on aspect abstractions that work as a strategy for recovering software design. By using our approach it is possible to employ general purpose aspects that represent software layers. This is useful for capturing such design in systems where a layered architecture exists but was not documented or where it has been inconsistently translated from design to code. The pattern is a generalization of our initial validation performed in a case study on the Open Service Gateway Initiative (OSGi) service platform. We could verify that its software layers are well defined in the specification and design, however when analyzing the actual Application Programming Interface (API), such layers are completely scattered over interfaces that inconsistently accumulate roles from different layers. By extracting the layered design into separate aspects, we were able to better understand the code, as well as explicitly identifying the affected layers when applying dependability crosscutting concerns to a concrete aspect solution on top of three different implementations of the OSGi platform.**

Keywords-Software architecture; Software layers; Software reengineering; Aspect-oriented programming.

I. INTRODUCTION

Reverse engineering, *Reengineering* and *Restructuring* are close terms, with subtle differences. Definitions from [3] indicate reengineering as the examination and alteration of a system to reconstitute it to a new form, while restructuring consists of transforming the system code keeping it at the same relative abstraction level, and preserving its functionality. Reverse engineering would consist of analyzing a system in order to identify its components and to create abstract representations of it.

Recovering lost information (e.g., design) and facilitating reuse are important reasons for reengineering [3]. Other reasons [4] leading to reengineering a software system are: insufficient documentation, improper layering, lack of modularity, duplicated code or functionality are among the coarse-grained problems. As a part of the reengineering process, one may employ techniques, such as refactoring [7] as a form of code restructuring. Refactoring consists of "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure". Aspect-oriented programming (AOP) [15] is a paradigm that is very useful when restructuring and reengineering systems. It allows changing the system without actually changing the system's source code. It is possible to keep cross-cutting concerns separate from the target system code at development time. Such concerns can be later integrated by "weaving" them to the target application either at compile time or during runtime.

Typical usages of AOP are straightforward solutions that either refactor crosscutting concerns out of the system code or introduce crosscutting concerns in the form of aspects woven in the system. Sometimes it may not be clear which system layers are being crosscut by which aspects, especially in systems with weak design or where the implemented design differs from documentation. In this paper, we propose the usage of aspects for providing another level of indirection that helps understanding systems that are reengineered with AOP. We provide an AOP refactoring pattern that helps capturing system design by aspectizing software layers, which can be reused by aspects that are applying concrete aspects as concerns that crosscut such layers (and consequently the system). Such abstractions we propose are useful for better understanding software architectures in systems with weak design (e.g., monolithic systems) or where design has been badly translated from the specification during its implementation. Therefore, the contributions of this paper are: 1) an approach for using aspects as an abstraction for capturing lost architectural design; 2) the refactoring of specific aspects that will target such abstractions instead of coding the aspects directly against the target system code; and 3) a reengineering pattern that guides through the process of extracting such design.

The pattern described here was validated in a case study of the OSGi [17] service platform. We present an architectural perspective that is useful in the context of reverse engineering for recovering lost design information, as well as in the context of reengineering when applying changes to the system and reusing the definitions of such abstractions that recover lost design. The remainder of this paper is organized as follows: Section 2 provides an overview of the problem, Section 3 describes the reengineering pattern, Section 4 details the case study in the OSGi platform, Section 5 discusses related work, followed by Section 6 that concludes this article.

II. OVERVIEW

Software layers [1] are an architectural pattern extensively used for grouping different levels of abstraction in a system. By employing such pattern for layered architectures, it is a good practice to design a flat interface that offers all services from a given layer. In a purist layer design, a layer of a system should only communicate with its adjacent layers, via such flat

 ISBN: 978-1-61208-438-1

interfaces. Such type of design gives a commonly used architectural view of systems. We find cases where the system is well designed in terms of layering, but the corresponding implemented code does not represent explicitly such architecture. In other (worst) cases, the system lacks good abstraction during design, resulting in a monolithic architecture which is hard to understand.

From now on in this article, the term reengineering will be employed as a general task – which may involve reverse engineering and restructuring – for improving system code and design. By reengineering the code, it is possible to arrive at a system whose architecture is more transparent, and easier to understand. In [4], extracting the design is considered as a first step for performing new implementations. Either if reimplementing the system or just applying the required changes, this step is very important.

A. Aspect-oriented Programming

The principle of Aspect-oriented Programming [15] is a paradigm that improves the modularity of applications by employing the principle of Separation of Concerns (SoC) advocated by Dijkstra [6]. In SoC, one should focus on one aspect of a problem at a time, as a way to have a better reasoning on a specific aspect of a system. An aspect should be studied in isolation from the other aspects but without ignoring them.

Putting these concepts into practice, AOP allows the separation of concerns (e.g., logging, transactions, distribution) that crosscut different parts of an application. These crosscutting concerns are kept separate from the main application code instead of being scattered over different parts of the system, as illustrated in Figure 1. A source file (e.g., module, class) may also have code that accumulates different responsibilities not necessarily related, giving an impression of tangled code.

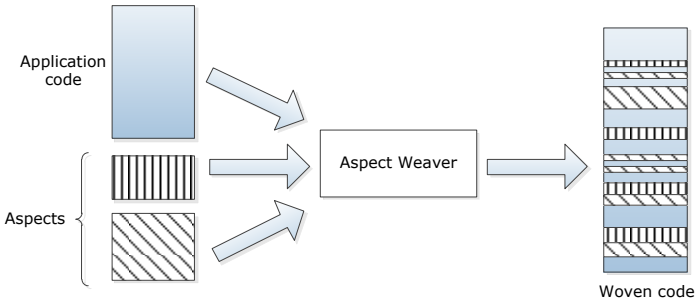


Figure 1. Illustration of how aspects are maintained outside the target application code, and then are intermixed with it.

AOP employs its own terminology, from which we briefly clarify some of the commonly used expressions that are going to be frequently cited throughout this article. Join *points* are constructs that capture specific parts of program flow (e.g., method call, constructor call). *Pointcuts* are elements that pick one or more specific join points in the program flow. The code that is injected into pointcuts during the weaving process is called advice in AOP terminology.

B. Lost Design

AOP is useful in the context of reengineering either to apply changes to code by introducing new crosscutting concerns, or by refactoring out from code existing crosscutting concerns into aspects. When in such AOP usage, we propose to give more semantics to pointcuts in a way that it is possible to represent part of the system design, by grouping the pointcuts in meaningful abstractions (e.g., layers) that could be reused. Our proposition does not involve changes in the aspect language level, but rather relies on existing constructs for building such abstract representations. Figure 2 illustrates an example where aspects are applied directly to the system code, and later layers are introduced as reusable aspects that contain more semantics.

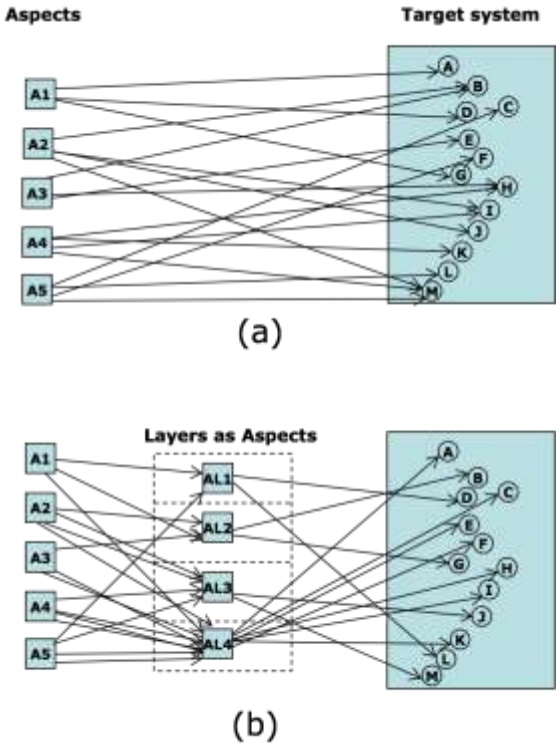


Figure 2. Aspects defining pointcuts (circles) on the reengineered system that are logically grouped in intermediary abstractions (layers as apects) that can be used to "visualize" the system's layers.

In a typical utilization of aspects, we define pointcuts using join points that directly reference the code of the target system, without any intermediary abstractions. This may end up with redundant pointcut definitions, especially in larger systems or in systems where aspects represent a significant part of the code. This redundancy is illustrated in Figure 2 by the pointcuts B, H, I and M, which are used by more than one aspect. If each definition involves several join points (e.g., method calls, method executions, instantiations), it may be difficult to give some reusable semantics to it. In addition, if the same set of join points is to be used in another aspect, we end up with redundant code. Indeed, we can give aliases to pointcuts for better expressiveness and reuse within the same aspect as we illustrate further.

C. Approach

At large, what we propose is to logically group pointcuts in general purpose aspect definitions that do not provide advices

 ISBN: 978-1-61208-438-1

```
public aspect A2 {
 void   around():   execution(void   Foo+.set*(..))
        || execution(void Bar.setFoo(Foo)){
    //advice code
 }}
public aspect A4 {
 pointcut   X():   execution(void   Foo+.set*(..))
          || execution(void Bar.setFoo(Foo));
 void around(): X() {
    //advice code
  }}
```

Figure 3. The same pointcut definition in the form of an anonymous pointcut in aspect A2 and as a named pointcut in aspect A4.

but only pointcut definitions. That gives more semantics to the aspects, allowing us to logically represent software layers that were not correctly (or not at all) represented in the original system. In the case from our example, the monolithic design of the target system is now represented with aspects that mimic layers (e.g., data access layer, GUI layer). They provide a new abstraction that captures such design concept. We also avoid redundant definitions of pointcuts. For instance, instead of aspects A2 and A3 having to write pointcut B twice, such pointcut is going to be logically grouped together with B in an aspect layer (AL2). The code from A2 and A3 can then reuse the pointcuts from AL2. After this change we now know explicitly that aspects A2 and A3 crosscut the layer represented by AL2. Another conclusion that can be drawn is that layer AL4 is crosscut by all aspects.

To clarify this proposition, we provide some code illustrating our approach. By taking the example of Figure 2 (a), the origin of the links toward the pointcuts (A through M, in the figure) denotes where the corresponding pointcut definitions are located. In such approach, it is normal to have the same pointcut definitions that may be present in different aspects, which represents redundant code as exemplified in Figure 3. The anonymous pointcut definition in A2 is the same used in A4 but cannot be reused, working as a sort of ad hoc pointcut. In contrast, the pointcut X of aspect A4 can be used by different advices just by referring to its name. Based on that reuse possibility, we suggest reusable pointcut definitions logically grouped, providing the semantics of a software layer.

In Figure 2 (b), our approach proposes the introduction of an intermediary abstraction that uses aspects for gathering cohesive pointcuts that would refer to join point in the same software layer. We can use these groupings to represent software layers and also to reuse the pointcut definitions with more semantics. Whenever reusing a pointcut, one would know which layer it refers to. In the example, each aspect layer (AL) illustrated will just group pointcut definitions (A to M) that belong to the same software layer, thus providing a representation of that layer as an aspect. The actual crosscutting concerns should be coded in aspects that refer to the pointcut definitions of these layer aspects, instead of repeating them in their code.

The code in Figure 4 that illustrates the layers is presented in Figure 2 (b) where we provide the example of the aspect layer

```
public aspect AL3 {
  pointcut J(): /* ... ... */
 pointcut    M():    execution(void    Foo+.set*(..))
        || execution(void Bar.setFoo(Foo));
}
public aspect A2 {
 void around():  AL3.M() {
        //code
  }
}
public aspect A4 {
 void around():  AL3.M() || AL4.K(){
        //code
  }
}
```

Figure 4. Layer aspect AL3 defines the redundant pointcut of previous example.

AL3, which represents an architectural layer (e.g., data access layer) that was "captured" using two pointcuts. The other two aspects of the example, A2 and A4, reuse the definition of the pointcut M. It is clear that both aspects A2 and A4 crosscut the layer represented by AL3. In the case of aspect A4, one can easily identify just by reading the code that it also crosscuts the layer represented by AL4. The illustrated advice of AL4 will be used whenever the program flow reaches the join points defined by pointcuts AL3.M or AL4.K.

III. PROPOSED PATTERN

A reengineering pattern is more related with the discovery and transformation of a system, than with the design structure [4]. It is important to note, however, that our proposed reengineering pattern describes a discovery process that involves the identification of a design element (an architectural pattern).

In the next subsections, we employ a similar organization (intent, problem, solution, tradeoffs) to the patterns defined in the Object Oriented reengineering patterns book [4] for describing our pattern named as *"Aspectize the Software Layers"*.

A. Intent

Utilizing reusable aspects for extracting the layered design of the system and clarifying where (and which) are such software layers.

B. Problem

Common usages of AOP are basically employed in two ways. The first one consists of refactoring crosscutting concerns out of the system code. The second case consists of introducing previously non-existent crosscutting concerns into the system, in the form of aspects. Both cases typically employ straightforward solutions that do not use intermediary abstractions. It is not clear which system layers are being affected (i.e., crosscut), especially in systems with weak design (e.g., monolithic systems) or where design has been badly translated from the specification during its

 ISBN: 978-1-61208-438-1

implementation. In larger solutions, pointcuts tend to be repeated where reuse could be possible. An extra level of indirection could introduce more semantics and pointcut reuse.

C. Solution

Introduce general purpose aspects (i.e., without advices) logically grouping correlated pointcuts, allowing to provide representations of the software layers used in the systems. The pointcut can be reused with better semantics than previously.

Before actually executing the necessary steps, it is important to understand the system being refactored. Applying some of the reverse engineering patterns defined in [4] can help:

- *Speculate about design*. It will allow making hypotheses about existing design so we are able to understand which ones are the existing layers.
- *Refactor to understand*. This is important to understand the code; even if these performed refactorings are not taken into account later (it might be the case when changing existing code is not desired).
- *Look for the contracts*. The proposed intent of this pattern is to infer the usage of class interfaces by observing how client code uses it. In the context of our pattern, this may be the case when contracts are not specific.

After identifying which are the layers and which to be abstracted, it is necessary to create their corresponding aspects. Each aspect will define the pointcuts that represent the services provided by a layer. The granularity level depends on the usage or what is necessary to be represented. For example, a data access layer abstraction could include pointcuts defining the general CRUD (create, read, update, delete) operations as the layer's services.

The layer aspects themselves do not provide any code for advices; therefore alone they are useless. The layer aspects should be reused by advices from other aspects that apply crosscutting concerns (e.g., logging, transactions, distribution) to the target system. In the case where such crosscutting concerns already exist in the form of aspects, it is necessary to apply the look for the contracts pattern in order to understand how these aspects use the target system. Wrapping the aspectized layers as a library that can be imported by the concrete aspects consists of a good reuse practice that should be employed whenever possible.

D. Trade-Offs

Following the format proposed in [4], the following trade-offs can be considered.

Pros:

- Higher level abstractions
- Clarification of the existing architecture through the extracted design
- Reusable pointcut definitions

Cons:

- Depending on the coverage of the aspects (e.g., crosscuts only parts of the system) the resultant design that was extracted may not completely describe the system architecture
- Poor knowledge of the system may also result in an incomplete representation

IV. CASE STUDY

Our initial validation of the proposed pattern was performed on the OSGi Service Platform [17], a dynamic environment where components may be installed, started, stopped, updated or unloaded at runtime. The API is standardized and the common point for different implementations. When aspects target the API they become applicable to any of the implementations. In the case of OSGi, we could verify this in our experiment involving the open source implementations of that API: Apache Felix, Equinox and Knopflerfish. We initially applied dependability aspects that were scattered over layers. Our approach used Aspect-J and the Eclipse IDE for defining and weaving the aspects in those implementations. An important fact to be pointed out is that the OSGi implementations in question did not use any aspect-oriented language prior to our intervention.

When we needed to identify which layers were being affected by which aspects, we could not easily tell because the way the specification presents the layers is much cleaner and less entangled that the reality in the API. The next subsections show the steps taken for applying our reverse engineering pattern.

A. Disentangling OSGi layers

As part of our analysis (speculate about design and look for the contracts) we have noted that useful concepts described in the OSGi specification are not well represented in its API, making it difficult to distinguish the layers in the specification from their counterparts in the API. The OSGi specification proposes a layered architecture, as depicted in the gradient boxes in Figure 5. The service, lifecycle, module and security layers are provided by the OSGi implementations, while the bundles layer represents the third party components that are deployed and executed on the OSGi platform. However, the software layers specified by OSGi are scattered over different interfaces, which accumulate roles from different layers.

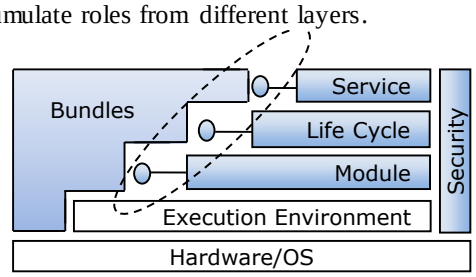


Figure 5. The proposed aspects simulate a single point of access (dashed ellipse) for each layer in OSGi's pseudo-layered architecture.

We found no single entity to describe individual layers in OSGi neither single access points for accessing the services provided by each layer. However, with such layer concept lost when a specification is translated into an API, we lose modularity as well. In the OSGi platform, the bundles layer freely accesses the other three layers (Figure 5). But, in practice, such access in OSGi is not done through a single interface per

layer. Actually, there is no such flat interface for explicitly representing layers in OSGi's API. The functionality of each layer is scattered over different interfaces which may accumulate roles from different layers. In our case study targeting OSGi we have employed our pattern for abstracting the Service, Lifecycle and Module layers, and then refactoring the dependability patterns to use it. We have not handled the security layer because it is an optional layer in OSGi implementations. The "aspectization" of the lifecycle layer (service and module layers were left out due to space limitations) is illustrated in Figure 6 and is a result of the next step when applying our pattern. The refactor to understand pattern was also helpful, and in our case it happened previously, at the time we applied the dependability aspects.

```
public aspect LifeCycle {

 pointcut install():
    execution(Bundle
BundleContext+.installBundle(String,..));

  pointcut stop():
    execution(void Bundle+.stop(..));

  pointcut start():
    execution(void Bundle+.start(..));

  pointcut uninstall():
    execution(void Bundle+.uninstall());

  pointcut update():
    execution(void Bundle+.update(..));

  pointcut resolve():
    execution(boolean
    PackageAdmin+.resolveBundles(Bundle[]));

  pointcut refresh():
    execution(void
   PackageAdmin+.refreshPackages(Bundle[]));

  pointcut activate():
    call(void
     BundleActivator+.start(BundleContext));

  pointcut deactivate():
    call(void
      BundleActivator+.stop(BundleContext));
}
```

Figure 6. Aspect representing OSGi's lifecycle layer

In Figure 6, the methods and transitions that concern bundle lifecycle are scattered across four interfaces (Bundle, BundleContext, BundleActivator, PackageAdmin) that already have roles other than lifecycle management. The different state transitions of a bundle's lifecycle are scattered over different interfaces. The install state transition is actually fired in the BundleContext interface. The resolve transition is defined in the PackageAdmin service interface, while the update and uninstall can be found in the Bundle interface. The refresh transition is part of the package admin, which is not part of the core API but rather declared in the PackageAdmin. The start and stop are both located in the Bundle and BundleActivator interfaces. In case of a Bundle having a BundleActivator, those calls are delegated to it. In the LifeCycle aspect we have rather called it as activation and deactivation, respectively.

A simple illustration of the lifecycle layer aspect being reused is shown in the advice from Figure 7, which provides a practical usage of an aspect targeting that layer by reusing the Lifecycle aspect (i.e., an aspect that abstract a software layer). The semantics of the code becomes clearer with a higher level concept. Although the original definition of the start pointcut involves only one join point in the Bundle interface, other cases that involve long pointcut definitions would gain more in terms of reuse and semantics gain.

```
public aspect ComponentIsolation {
void around(Bundle b): LifeCycle.start()
     && !cflowbelow(LifeCycle.start())&&
         this(b){
 if (!PlatformProxy.isSandbox() &&
     PolicyChecker.checkIsolation(b)){
     PlatformProxy.start(b.getBundleId());
  } else {
    proceed();
  }
 }
}
```

Figure 7. Example of layer aspect being reused

B. Discussion

Reverse engineering is a fundamental part of the reengineering process, since understanding the system is an important step before changing or reconstructing it. The usage of our pattern allowed us to recover lost information (the translated design) in OSGi, and also facilitated the reuse of that abstraction, thus achieving essential goals of reengineering. Although this article focuses only on one architectural pattern, software layers, we could illustrate how to use aspects to capture an architectural abstraction without needing to restructure the code, which in addition can be reused to apply other crosscutting concerns. The lack of an automated approach was a major drawback that required the manual analysis of the target system code. This would represent an obstacle for applying such approach in systems with significant size. Therefore, the development of auxiliary tools for applying that reengineering pattern would significantly improve the efficiency of using such approach.

V. RELATED WORK

Other approaches, such as [5][12][14][16] and [20], employed pattern-based reverse engineering, which consists of detecting design patterns in software. An important motivation for providing such mechanisms is that patterns provide a common idiom for developers. Therefore by understanding what patterns where employed, the effort necessary to understanding the whole software will be reduced [20]. These approaches help identifying the architectural elements based on the recognition of patterns. Although a method for software architecture reconstruction is discussed in [12], the process is based on design patterns recognition. In summary, most of the above strategies try to automate the lookup of the more traditional design patterns [9], with tools inferring patterns based on graph

 ISBN: 978-1-61208-438-1

analysis and visual tools showing such relationships and pattern match.

The work in [5] slightly differs from such approaches because it allows looking for anti-patterns and "bad smells" that may negatively affect the architecture recovery. In contrast to our work, although the previously mentioned approaches provide a sort of (semi-) automated discovery of patterns, they are rather focused on a fine grain perspective of patterns (i.e., design patterns), while we intend to employ a strategy that gives us a coarse grain perspective of an architectural pattern (currently limited do software layers).

The relationship between patterns and AOP that we found in literature mainly deals with the implementation of design patterns with the help of AOP [13], and studies that analyze impacts and drawbacks of such implementations [2][11][23]. Under the perspective of software architectures, for instance, some research efforts focused on establishing a way to represent aspects in the software architecture early in the design phase, using aspect-oriented architectural models [8] – sometimes identifying them even earlier, during the requirements elicitation phase [22] – and more specific forms of expressing them such as the definition of representations of aspects using the Unified Modeling Language (UML), as found in [18] and [25]. Another example in the architectural level is that of specific Architecture Description Languages [10][19][21] that are able to express aspects and other crosscutting concerns. However, the above cases of aspect usage focus on how to represent in the architecture the aspects that crosscut the system elements (e.g., components, modules, subsystems), while our approach uses an aspect abstraction to mimic an architectural pattern.

Specifically talking about the layers architectural pattern, the only study we have found explicitly dealing with software layers and AOP was performed in [24]. However, that report deals with software layers and aspects using a perspective that differs from our work. Their approach consists in the assessment of the impact of using AOP on layered software architectures.

VI. CONCLUSIONS AND FUTURE WORK

The reengineering of systems may be motivated by different reasons, such as lack of modularity, improper layering, duplicate code or functionality. Refactoring with the help of aspect-oriented programming provides a way of performing reengineering by employing the separation of concerns principle. It allows cross-cutting concerns to be separated from the application, allowing better maintenance and readability.

This article proposed the usage of aspects in a novel way. It was used to provide an abstraction that provides a correct perspective of a layer architecture that was not well represented in the system code. It refactors specific aspects in order to use such abstractions instead of targeting the system code, and we also propose a reengineering pattern describing such process.

In the case study, the usage of aspects allowed us to abstract logical layers that were scattered over the OSGi API, providing a vision that disentangles the OSGi layers from the interfaces and classes that accumulate responsibilities from different layers. The resulting aspectized layers were reused for applying concrete aspects that concerned dependability. Since analyzing a single case study may be a limitation of the generalized perspective provided in this article, for future work we plan to apply this pattern in other systems for evaluating its occurrence, as well as getting a deeper understanding of its advantages and drawbacks. Another interesting path to take is to evaluate how employ aspects to recover and represent other architectural patterns.

ACKNOWLEDGEMENT

This work was supported by INES - Instituto Nacional de Ciência e Tecnologia para Engenharia de Software (http://www.ines.org.br/) - with financial support from CNPq. The work presented here was initially carried out as part of the ASPIRE project, co-funded by the European Commission under the FP7 programme, contract 215417.

REFERENCES

[1] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture: A System of Patterns", Wiley, 1996, ISBN: 978-0-471-95869-7

[2] N. Cacho, C. Sant'Anna, E. Figueiredo, A. Garcia, T. Batista, and C. Lucena, "Composing design patterns: a scalability study of aspect-oriented programming". In Proceedings of the 5th international conference on Aspect-oriented software development. ACM, 2006, pp. 109-121, ISBN:1-59593-300-X, doi:10.1145/1119655.1119672

[3] E. Chikofsky and J. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy". IEEE Software 7, 1, January 1990, pp. 13-17, ISSN:0740-7459, doi:10.1109/52.43044

[4] S. Demeyer, S. Ducasse, and O. Nierstrasz, "Object Oriented Reengineering Patterns". Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002, ISBN: 978-3952334126

[5] M. Detten and S. Becker, "Combining clustering and pattern detection for the reengineering of component-based software system" Joint ACM SIGSOFT conference -- QoSA and ACM SIGSOFT symposium -- ISARCS on Quality of software architectures -- QoSA and architecting critical systems. ACM, New York, NY, USA, 2011, pp. 23-32, ISBN: 978-1-4503-0724-6, doi:10.1145/2000259.2000265

[6] E. Dijkstra, "On the role of scientific thought", EWD 447, 1974, appears in E.W.Dijkstra, Selected Writings on Computing: A Personal Perspective, Springer Verlag, 1982

[7] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, "Refactoring: Improving the Design of Existing Code". Addison Wesley, 1999, ISBN: 978-0201485677

[8] R. France, I. Ray, G. Georg, and S. Ghosh, "Aspect-oriented approach to early design modeling" Software, IEE Proceedings-Vol. 151, No. 4, 2004, pp. 173-185, doi: 10.1049/ip-sen:20040920

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, Menlo Park, CA, 1995, ISBN: 978-0201633610

[10] A. Garcia, C. Chavez, T. Batista, C. Sant'Anna, U. Kulesza, A. Rashid, and C. Lucena, "On the modular representation of architectural aspects" Software Architecture, Springer Berlin Heidelberg, 2006, pp. 82-97, ISBN 978-3-540-69272-0 doi: 10.1007/11966104_7

 ISBN: 978-1-61208-438-1

[11] A. Garcia et al. "Modularizing design patterns with aspects: a quantitative study" Transactions on Aspect-Oriented Software Development I, 2006, pp. 36-74, ISBN: 1-59593-042-6 doi:10.1145/1052898.1052899

[12] G.Y Guo, J. M. Atlee, and R. Kazman, "A software architecture reconstruction method". In Proceedings of the TC2 First Working IFIP Conference on Software Architecture (WICSA1), Kluwer, The Netherlands, 2006, pp. 15-34, ISBN:0-7923-8453-9

[13] J. Hannemann and G. Kiczales, "Design pattern implementation in Java and AspectJ". Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '02), ACM SIGPLAN Notices, 2002, pp. 161-173, ISBN:1-58113-471-1 doi:10.1145/582419.582436

[14] R. K. Keller, R. Schauer, S. Robitaille, and P. Page, "Pattern-Based Reverse-Engineering of Design Components" 21st International Conference on Software Engineering, IEEE Computer Society Press, May 1999, pp 226–235, ISBN: 1-58113-074-0

[15] G. Kiczales, G., et al. "Aspect-Oriented Programming" European Conference on Object-Oriented Programming (ECOOP), Springer-Verlag, Finland, 1997, ISBN: 978-3-540-69127-3, doi: 10.1007/BFb0053381

[16] J. Niere, W. Shafer, J. P. Wadsack, L. Wendehals, and J. Welsh, "Towards pattern-based design recovery" International Conference on Software Engineering, IEEE Computer Society Press, May 2002, pp. 338–348, ISBN: 1-58113-472-X

[17] OSGi Alliance. OSGi Service Platform. http://www.osgi.org [retrieved: 09, 2015]

[18] R. Pawlak, L. Duchien, G. Florin, F. Legond-Aubry, L. Seinturier, and L. Martelli, "A UML notation for aspect-oriented software design" Proceedings of the AOM with UML workshop at AOSD, Vol. 2002.

[19] N Pessemier, L. Seinturier L., T. Coupaye, and L. Duchien, "A model for developing component-based and aspect-oriented systems" Software Composition. Springer Berlin Heidelberg, 2006, pp. 259-274, ISBN: 978-3-540-37659-0, doi: 10.1007/11821946_17

[20] I. Philippow, D. Streitferdt, M. Riebisch, and S. Naumann, "An approach for reverse engineering of design patterns" Software Systems Modeling, 2005, pp. 55–70, ISSN: 1619-1374 , doi: 10.1007/s10270-004-0059-9

[21] M. Pinto and L. Fuentes, "AO-ADL: An ADL for describing aspect-oriented architectures". Early Aspects: Current Challenges and Future Directions, 2007, pp. 94-114, doi:10.1007/978-3-540-76811-1_6

[22] A. Rashid, P. Sawyer, A. Moreira, and J. Araújo, "Early aspects: A model for aspect-oriented requirements engineering" IEEE Joint International Conference on Requirements Engineering, 2002, pp. 199-202, ISSN: 1090-705X, doi: 10.1109/ICRE.2002.1048526

[23] C. Sant'Anna, A. Garcia, U. Kulesza, C. Lucena, and A. V. Staa, "Design patterns as aspects: A quantitative assessment" Journal of the Brazilian Computer Society, 10(2), 2004, pp. 42-55, ISSN: 1678-4804, doi: 10.1007/BF03192358

[24] J. Saraiva, F. Castor, S. and Soares, "Assessing the Impact of AOSD on Layered Software Architectures" ECSA 2010, LNCS 6285, 2010, pp. 344–351, doi: 10.1007/978-3-642-15114-9_27

[25] J. Suzuki and Y. Yamamoto, "Extending UML with aspects: Aspect support in the design phase" Lecture Notes in Computer Science, Springer-Verlag London, UK, 1999, pp. 299-300, ISBN:3-540-66954-X

 ISBN: 978-1-61208-438-1

Networking-based Personalized Research Environment : NePRE

Heeseok Choi, Jiyoung Park, Hyoungseop Shim, Beomjong You
Division of Advanced Information Convergence
Korea Institute of Science and Technology Information
Daejeon, South Korea
email: {choihs, julia.park, hsshim, ybj}@kisti.re.kr

***Abstract*—With advancement in information technologies and a better mobile environment, the paradigm of service is shifting again from web portals to individual applications based on any network. On the other hand, as more investment is being made in R&D, the efforts to enhance R&D productivity are becoming important. This study proposes a service model of Networking-based Personalized Research Environment (NePRE) for developing the tool to assist researchers in their R&D efforts. It can be easily utilized by researchers in their R&D information activities. To do this, we compare services and tools in terms of information activities in R&D. And we also analyze changes of information environment in terms of personalization. Subsequently, we design a service model of NePRE. Finally, we define its key functions to assist researchers with respect to their six information activities in R&D life-cycle.**

Keywords-research support; persoanlization; R&D life-cycle.

I. INTRODUCTION

With advancement in information technologies and a better mobile environment, the paradigm of service is shifting again from web portals to individual applications based on any network. On the other hand, as more investment is being made in R&D, efforts to enhance R&D productivity of researchers are becoming more important. Light-weight applications are already being developed and disseminated to assist researchers in their R&D efforts. Unfortunately, however, the utilized data are limited to overseas database with a weak linkage to domestic academic information resources. Furthermore, it is not still easy to perform information-aid R&D since users have to access each service individually. In addition, recent changes to information environment makes personalized service more important for convenient information usage by researchers.

This study proposes a service model of networking-based personalized research environment for developing the tool to assist researchers in their R&D efforts. It can be easily and conveniently utilized by researchers in their R&D information activities.

This paper is organized as follows. In Section 2, we describe existing science and technology information services, and discuss changes in information environment. Section 3 introduces our design of a networking-based personalized research environment. Subsequently in Section 4, we define six key informative functions to assist researchers in R&D. Finally in Section 5, we discuss conclusions and future works.

II. RELATED WORKS

A. Existing Science and Technology Information Services

Services and tools were already developed and being made good use of assisting researcher's R&D activities. They provide useful information ranging from papers, patents or other academic information to that on projects or researchers. Furthermore, they also offer research support features through bibliographic information management. Representative services and tools are as follows.

- SciVal Suite [1]: This provides a critical information about performance and expertise to help enable informed decision-making and drive successful outcomes. It is composed of SciVal Spotlight, SciVal Strata, and SciVal Experts. It helps decision makers responsible for research management to assess institutional strengths and demonstrable competencies within a global, scientific landscape of disciplines and competitor. And it helps decision makers to identify researcher expertise and enable collaboration within the organization and across institutions. And it also helps them to measure individual or team performance across a flexible spectrum of benchmarks and measures.
- Mendeley [2]: This is a reference manager and academic social network. It makes your own fully-searchable library in seconds, cite as you write, and read and annotate your PDFs on any device. It manages bibliographic information, and helps researchers generate references when they write them in a paper. In addition, it helps finding collaborative researchers of the world, and supports composing community with them.
- RefWorks [3]: This is an online research management, writing and collaboration tool. It is designed to help researchers easily gather, manage, store and share all types of information, as well as

 ISBN: 978-1-61208-438-1

generate citations and bibliographies. If researchers need to manage information for any reason whether it is for writing, research or collaboration, RefWorks is the good tool.

- National Digital Science Library (NDSL) [4]: This is an integrated science and technology information service including paper, patent, and research reports by KISTI of Korea. It provides the specialized search service and integrated search to 0.1 billion of contents. It promotes efficient access to quality science and technology information based on cooperation networks.
- National Science and Technology Information Service (NTIS) [5]: This has been built as a national R&D knowledge portal for providing information regarding national R&D projects in connection with each ministry and institution. In NTIS, standard metadata are connected and managed by systematically, which needs joint use thereof in the cross-ministerial level, for example, avoiding redundant similar projects in advance. Each ministry builds a system which supports real project management for the process from receiving R&D projects to outcome management, connects and just provides standard information to the NTIS.
- Research Information Center (RIC) [6]: This is a virtual research environment being jointly developed by the Technical Computing Group at Microsoft and The British Library. The purpose of the RIC is to support researchers in managing the increasingly complex range of tasks involved in carrying out research. Specifically, to provide structure to the research process, easy access to resources, guidance and tools to manage information assets, along with integrated collaboration services.
- ResearchGate [7]: This allows researchers around the world to collaborate more easily. It discovers scientific knowledge, and makes your research visible. For a common purpose of advancing scientific research, it links researchers from around the world. It is changing how scientists share and advance research in digital age.

Nevertheless, overseas bibliographic management tools lack linkages and utilization of diverse academic information resources, while domestic information services still remain at information search for R&D activities and don't have sufficient means for sharing individual information resources.

B. Changes in Information Environment

Recent information environment can be considered in terms of service personalization as follows.

- Open expansion of information and data: the demand for publicly available information and data is increasing due to government's 3.0 and activation policies of creative economy in Korea. Furthermore, there are more projects for publishing and sharing public data. In addition, there are an increasing number of data standardization as Linked Open Data (LOD) and LOD construction.
- Enhancement of personal information protection: people are more aware of protecting personal information and leakages of personal information in terms of the society and technology. Therefore, regulations and institutions are improved for enhancing personal information protection, and people involved therein have studied how to further enhance the technology. In particular, as the Personal Information Protection Act is enforced since 2011, collecting, using, providing, processing and managing personal information is strictly regulated to minimize personal information leakage. In addition, because increasing open and shared data contribute to combining and integrating the data to form information to identify personal identity, more efforts are required to protect personal information.
- Popularization of social networking service: a popular trend is currently to make a connection between online users about common subjects to share and use information and knowledge. Various social networking services and platforms, for example, facebook, youtube, twitter, Kakaotalk, LinkedIn, and ResearchGate are now used. The outlook is that they are connected with web portals or mobile services to further enhance social networking services.
- Very big contents: big data and IoT (Internet of Things) technology is developing fast, and services using them are appearing. Data are now more abundant and diversified than before, and non-literature data as well as literature-centered data will be more importantly handled.
- Advances in web platform technology: as web technology develops, services in various formats have been developed and distributed, for example, mobile apps and web apps. In particular, web-based application S/W based on web standard HTML 5 is even more valued, that can be installed and used in all devices from smartphones to smart TVs where web browsers operate.

Table 1 summarizes implications and direction toward good services when we look into changes of information environment in terms of service's personalization. That is, linking opened and standardized data is more important than directly constructing many contents. Also personal services should depend on personal participation rather than collecting personal information. For contents curation from very big contents, social networking is becoming more important in order to utilize group intelligence. Since web applications are based on the web, they can be operated just by web browsers. That is, web applications can be easier than web portals for personal usage on any device.

 ISBN: 978-1-61208-438-1

TABLE 1. CHANGES OF INFORMATION ENVIRONMENT

Division	Implication	As-Is→To-Be
Open expansion of information and data	.useful contents are more plentiful and various .link and usage of open contents are possible .link is possible by standardized methods(API, LOD)	Construction →Link
Enhancement of personal information	.collection of personal information from web become difficult . construction of personal profile information in web is difficult	Usage of personal information→ Personal partiipation
Popularization of social networking service	.information link is possible and important .share of information is important .easy collaboration .participation of community is important	Personal intelligence → Group intelligence
Very big contents	.curation is important .topic-based information link and statistical analysis are important .variety and instantaneity of useful contents	Search → Analysis
Advances in web platform technology	.services independent from devices and web browsers is important . resources usage of cloud environment is important	Web portal → Web application

With open expansion of information and data, enhancement of personal information is a challenging status to the personalization of services. Fortunately, social networking services were already popular over the world, and they were enhanced to communicate and to collaborate each other on any subject. Therefore, social networking can be used to realize personalization of a service.

III. DESIGN OF A NETWORKING-BASED PERSONALIZED RESEARCH ENVIRONMENT

We first established three views of personalized research environment. First of all, in the function view, functions are defined through how information is used to retrieve, collect, analyze, collaborate, store and writhe outcomes - in the R&D process [8][9][10] from the step of ideas & planning to the step of outcomes. The functions can be summarized as in Table 2.

Next, the contents view defines what information can be utilized concerning service functions required in a relevant R&D process. In order to satisfy the requests listed on Table 2, it is essential to share and use information resources held by individual researchers in addition to domestic and overseas scientific technology information. Individual knowledge tools should allow users to utilize various information resources - involving domestic and overseas information resources and individually held one.

TABLE 2. FUNCTION VIEW OF SERVICE

Activities	R&D Activities	Function requirements
Search	.Identification of research trends, core patents .identification of research-related topic and concept .discovery of research topic .store of academic search results	.Categorization and management of searched results .Expert recommendation .recommendation of research topic .categorization and management of academic search results
Collect	.Collection of bibligraphy and its original literature	.Auto-management of bibliographic information .recommendation of materials related to concerning topic .articles viewing based on bibliometric information
Analyze	.analysis of technology ripple effect .Statistical analysis .analysis of citation relation among technology groups	.Technology trend, topic analysis .Data statistics .provision of relation map
Collaborate	.Researcher network analysis .Management of personal R&D profile	.data share with collaborative researchers .construction and collaboration of community .writing memos in document and sharing them
Store	.store of academic information resources	. categorization and management of academic information resources
Publish	.Organization of research results .writing papers	.Support writing document based on template

Table 3 summarizes the contents that NePRE manages and uses for supporting researchers.

TABLE 3. CONTENTS VIEW OF SERVICE

Division	Description
External resources	paper, research report, patent, fact information, R&D project information, standard, trend/analysis information, bibliographic information, organization information
Internal resources	paper, research report, patent, memos, images, personal profile, web resources

Finally, the operation view defines how to link and take advantage of information resources in individual knowledge tools. Table 4 shows the operation view. It defines data

 ISBN: 978-1-61208-438-1

categorization and relationships based on bibliographic information such as patent and research reports. The tools are designed in a structure to offer information assisting R&D activities and at the same time to get feedbacks of and save information resources created in that process.

TABLE 4. OPERATION VIEW OF SERVICE

Division	Description
Platform-based	Link overseas and domestic S&T information resources by using KISTI science and technology knowledge platform as hub
Biobliometric-based	Categorize, store, search by using bibliographic information Such as paper, patent, research report
Bi-directional link	Provide, store, and manage bidirectional information between tool and researchers

IV. SIX KEY FUNTIONS IN NEPRE

In this paper, we present conceptual model and design principles on a personal research environment that can be easily used with installation on various device environment of each individual researcher in Figure 2. Researchers perform various R&D information activities that are 'search', 'collect', 'analyze', 'collaborate', 'store', 'publish' in R&D process.

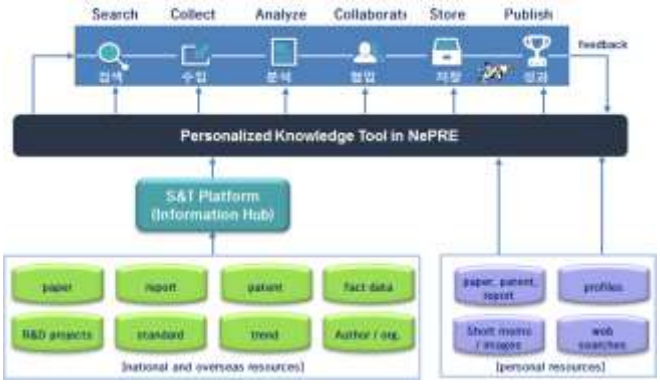


Figure 1. Conceptual model of personalized research environment

Both opened S&T information resources and researchers' personal resources can be converged and used. That is, it is essential to integrate and use individual researcher's information resources as well as Korea's and overseas science and technology information resources in order to facilitate various information activities carried out in R&D by researchers. The aforementioned personal knowledge tool must be able to be installed and operated in various types of device environment preferred by each researcher [11]. In addition, they must be connected in a standardized manner to use national and overseas information resources.

The NePRE provides a collection of functions to support the R&D life-cycle of researchers as follows.

- **Search**. The NePRE provides researchers with a easy access to better resources based on statistical analysis of citation. For example, the resources having many references from other resources are selected preferentially, or the resources strongly referenced by similar researchers are selected. The search operation executes contents curation over simple search via participation of other researchers. The contents curation finds high quality of resources that can be more reliable using similar researchers' experience while simple search just finds the resources closely matched to input keyword.
- **Collect**. A number of information resources have been already identified due to national and international policy of data open. The NePRE provides means to gain access to and leverage content. The user can automatically receive updates from specific resources sites, using e-mail and RSS feeds. Good articles can be recommended or identified by some researchers involved to similar topics. This suggests that open and shared resources are managed centered on individual user. The NePRE provides a link to the content supplier's site for share. The underlying content sources are defined as the Korea Institute of Science and Technology Information, academic community, publishers, or third party. All users would have subscriptions to all commercial resources likely to be available.
- **Analyze**. The NePRE provides an insight about technologic trends, competition relationship, and promising technologies. The NePRE supports analysis of technology's ripple effect, researcher network, convergence relationships between technology groups. And the NePRE helps us finding research topics, identify key patents, and understand status of technology development based on technology keywords. In addition the NePRE provides opinions of feasibility or necessity of any research.
- **Collaborate**. The NePRE is designed to make it easy to identify potential collaborators, create community, and share the results of other researcher like other tools [7][12]. Both national and international information resources are outlined and summarized via the participation of other researchers. The NePRE encourages researchers to work together to develop and populate their research results like RIC and ResearchGate. For collaborative research, community participation and enhancements are very important. Therefore, the NePRE provides means to encourage communities such as following researchers, following research, categorization for community customized services, sharing calendars per community, and personalization via a following function like SNS.
- **Store**. The NePRE offers an enhanced function to allow the researcher to save searches on cloud environment for seamless usage from any device.

 ISBN: 978-1-61208-438-1

The NePRE supports clipping or scrapping searched results. The cloud storage stores metadata associated with the documents, together with links to the full-text where permitted. When the resources are stored on a cloud environment, they are automatically related other resources each other and identified to be unique. The user can also easily access to all resources across various kinds of devices. Each user accesses to individual storage on cloud environment via an online account. Manage information resources. In addition, each user accesses to the resources of third party or local databases. It categorizes resources according to personal criteria.

- **Publish**. The NePREl supports publication life-cycle, from literature search and retrieval, papers annotation, and bibliography management to self – archiving like RIC [6]. The NePRE supports making template-based document such as papers, patents, and research reports. The NePRE supports automatically managing and listing up references, and helps researchers finding sources of resources. The NePRE supports writing short memos within documents and it helps share them with colleagues or in community.

V. CONCLUSION AND FUTURE WORK

This study suggests a service model of networking-based personalized research environment for developing personal knowledge tools researchers can use easily in their R&D. To do this, we first compared services and tools in terms of information activities in R&D. And we also analyzed changes of information environment in terms of personalization. In the suggested model, functions required for each information use type in R&D are defined. Contents concerned are extended and defined to integrate and use individual researchers' information resources as well as national and overseas science and technology information resources. The connection focusing on personal tools, not web portal-centered connection, is employed, and the method of operation is defined to facilitate connection and integration of information by using bibliography information of various information resources. Finally, we presented the outlook of six key informative functions of NePRE in R&D life-cycle.

In the future, we will compare NePRE to other tools through case study. In addition, future studies will focus on establishing a method of efficient connection and use of science and technology information resources by means of personal knowledge tools. It is necessary to study how to facilitate efficient classification and storage of individual researchers' information resources, and integration with connected data. It is necessary to study how to design light-weight personal knowledge tools.

REFERENCES

[1] Elsevier SchVal Suite, www.info.scival.com, August, 2015.

[2] Mendeley, www.mendeley.com, August, 2015.

[3] RefWorks, www.refworks.com, August, 2015.

[4] NDSL, www.ndsl.kr, August, 2015.

[5] NTIS, www.ntis.go.kr, August, 2015.

[6] R. S. Barga, "A Virtual Research Environment (VRE) for Bioscience Researchers", International Conference on Advanced Engineering Computing and Applications in Sciences, 2007, pp.31-38.

[7] ResearchGate, www.researchgate.net, August, 2015.

[8] H. Kim, N. Kwon, E. Jung, J. Lee, H. Choi, "Korean Scientists' R&D Life-Cycle Study", KISTI knowledge report, 2011.

[9] Y. Y. Yao, "A Framework for Web-based Research Support Systems", Proceedings of the 27th International Computer Software and Applications Conference (COMPSAC), 2003, pp.1-6.

[10] S. Kim and J. Yao, "Mobile Research Support Systems", 26th IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), 2013, pp.1-5.

[11] JISC briefing paper, "Digital Information Seekers", www.jisc.ac.kr/publications/reports/2010, August, 2015.

[12] D. D. Roure, "myExperiment: Defining the Social Virtual Research Environment", 4th IEEE International Conference on Science, 2008, pp.182-189.

 ISBN: 978-1-61208-438-1

Decision Making and Service Oriented Architecture for Recruitment Process. Using the New Standard Decision Model and Notation (DMN)

Fatima Boumahdi
LRDSI Laboratory, Sciences Faculty
Saad Dahlab University
BP 270 Soumaa Road Blida, Algeria
Email: F_boumahdi@esi.dz

Houssem Eddine Boulefrakh
University Mouloud Mammeri
Tizi Ouzou, Algeria
Email: h_boulefrakh@esi.dz

Rachid Chalal
LMCS Laboratory
Higher National School of Computer Science
ESI, Oued-Smar (Algiers), Algeria
Email: r_chalal@esi.dz

***Abstract*—Various models and methods are used to support the design process of SOA (Service Oriented Architecture), but still after many years of practice, there are a lot of questions and unsolved problems that cause the failure of SOA development projects. One of the reasons is that rapid changes in the business environment make it necessary to introduce the decision design, which should be efectively supported by SOA. Indeed, it is a big challenge to create a system that help the human resource development in industry to make their work easier without missing an opportunity to get a best employee. The objective of this study is to develop a decision making and Service Oriented Architecture for employee recruitment using analytic hierarchy process. This study explored the relationship between SOA and decision making during the recruitment process. To achieve our goal, we use the new method SOA^{+d} to develop the SOA architecture. Also we provide the decision using the new standard Decision Model and Notation (DMN). The novelty of the proposed approach is in the a) the formal definition of a complete set of proposed services b) the uses of standards language and notation in each dimension of approach c) the specification of the mappings rules to identify a set of services. We illustrate the proposed approach with a real case study of the Recruitment and Selection in SAAD DAHLAB University.**

Keywords–SOA; SOA^{+d}; DMN; SoaML, AHP.

I. INTRODUCTION

While organizations are trying to become more agile to better respond the market changes, and in the midst of rapidly globalizing competition, they are also facing new challenges. It is primarily a question of ensuring the decisional aspect of the information system by adopting the services oriented architecture (SOA) like a support architecture.

Also,the Human Resources Area needs to carry out different activities in order to find a person with the skills, abilities, experience and knowledge to fill a vacancy. This process is usually time-consuming whereby a lot of manual work is required and it is necessary to coordinate many people in the different stages of the process. The Recruitment and Selection process covers:

- Requesting a person with certain skills and abilities to fill a vacancy.
- Advertising the vacancy internal and external.
- Scheduling psych technical test, interviews, medical exams, etc.
- Collecting result of test and interviews.
- Updating the candidate list.

The decision environment consists of what a basic interview necessitates. The interviewer is the major decision-maker who chooses the right candidate for the vacancy. Decision period depends mainly on the corporate needs. The immediacy of the need of an employee, the time needed to fulfill the procedural requirements of the recruitment process together with the time that the interviews take (this may change depending on the number of the candidates) are the major factors that shape the decision period [1].

Additionally, in order to enable a vendor independent formalization of decision designs with a common understanding and tool support, the *Object Management Group* (OMG) decided to work on a standardized meta-model and a profile that enables the modeling of decision making and their elements [2]. The result of this effort is the Decision model notation *Decision Model and Notation* (DMN), which is currently released in version 1.0. Today, DMN gains increasing tool support, even IBM decided to integrate the DMN in their proprietary IBM Blueworks [3]. Therefore, the proposed architecture uses this new standard for modeling the decision view, and present the principal contribution of our work in decision field.

To develop a Service Oriented Architecture for the recuitement process, we must use the new approach SOA^{+d} [4] which integrates a decision aspect in SOA.

The SOA^{+d} approach comprises four phases [4] (a) Analysis phase: it contains three activities, each activity presents a view of SOA^{+d} and supported by standard modeling that reinforces key view. Therefore, the UML standard is used to analyse IS level. We use the BPMN standard for the business analysis, and we use the new DMN standard to specify decision view. (b) Identify and categorize services: in this phase, the applicable mapping rules for service identification based on the use case, BPMN and DMN are defined. (c) The service design phase,, in which a set of service designs has to be designed and modeled using the standard *Service oriented architecture Modeling Language* (SoaML). d) The realization services phase present the realization services using existing tools.

 ISBN: 978-1-61208-438-1

A. Problem statement

Given the conventional technique of interviewing for a vacancy in a company, the need of a more systematic approach is obvious. There are some criteria to be met while the decision is made. The recruitment decision should be based on:

- A consistent set of satisfaction of the requirements
- A clear and objective decision making environment
- A well defined and documented list of the requirements expected to be met by the candidates.

The main goal of our research is to design a new SOA to support the decision making in the recuitement process. In order to solve the main problem and develop a research plan of action, the following sub-problems were identified:

- What recruitment and selection strategies are suggested in the literature?
- To what extent does old methods utilise the recruitment and selection of strategies suggested in the literature?
- How the competent are sales managers in using the suggested recruitment and selection tools?

The recruiting process is typically intended to realize the following among other objectives:

- To provide an equal opportunity for potential candidates to apply for vacancies.
- To systematically collect information about each applicants ability to meet the requirements of positions.
- To attract highly qualified individuals.
- To select candidates who will be successful in performing the tasks and meeting the responsibilities of the position.
- To emphasize active recruitment of traditionally underrepresented groups, i.e. individuals with disabilities, minority group members and women in order to resolve historical recruitment imbalances.

B. Paper Organization

In Section II, the DMN notation (Section II-A) and Analytical Hierarchical Process (Section II-B) are presented. These concepts are used in our work for defining the new architecture. In Section III, we illustrate the different phases of SOA^{+d} approach. Section IV presentes the development of the case study by using the New standard DMN and SOA^{+d} in Recuitement process. Finally, we conclude the paper by proposing some future works (Section V).

II. BACKGROUND

In order to ensure comprehension, the following terms, related to this study, are briefly defined.

A. Decision Model and Notation

The OMG has recently standardized the DMN, which enables the abstract formalization of decision designs [5]. The goal of DMN is to standardise notations (and associated metamodel) for decision modelling. DMN will provide constructs spanning both decision requirements and decision logic modeling.

The decision requirements level consists of a Decision Requirements Graph (DRG) depicted in one or more Decision Requirements Diagrams (DRDs). A DRG models is a domain of decision making, that shows the most important elements involved in it and the dependencies between them [5]. The elements modeled are decisions, areas of business knowledge, and the input data.

Decision logic level : The components of the decision requirements level of a decision model may be described, as they are above, using only business concepts. This level of description is often sufficient for business analysis of a domain of decision-making, to identify the business decisions involved, their interrelationships, the areas of business knowledge and data required by them, and the sources of the business knowledge [5]. Below are the reasons of using a DMN:

- DMN creates a standardized bridge for the gap between the business decision design and decision implementation [5].
- DMN, as an IT specification, is a confirmation that there is demand for a new kind of software product aimed at decision modeling and management.
- Common notation that is readily understandable by all business users, from the business analysts needing to create initial decision requirements and then more detailed decision models, to the technical developers responsible for automating the decisions in processes, and finally, to the business people who will manage and monitor those decisions.

B. Analytical Hierarchical Process

AHP is a method for ranking decision alternatives and selecting the best one when the decision maker has multiple criteria [6]. It answers the question, Which one?. With AHP, the decision maker selects the alternative that best meets his or her decision criteria developing a numerical score to rank each decision alternative based on how well each alternative meets them.

The application of the AHP to the complex problem usually involves four major steps:

- **Step 1** : Break down the complex problem into a number of small constituent elements and then structure the elements in a hierarchical form.
- **Step 2**: Make a series of pair wise comparisons among the elements according to a ratio scale.
- **Step 3** : Use the eigenvalue method in order to estimate the relative weights of the elements.
- **Step 4** : Aggregate these relative weights and synthesize them for the final measurement of given decision alternatives.

III. SOA^{+D} APPROACH

SOA^{+d} method based on SoaML and DMN will be used. In the literature, several authors proposed approach of the *Service Oriented Architecture* (SOA) services [7]–[15]. According to the followed vision, each one proposes a set of steps. SOA^{+d} proposes a new approach for the development of the SOA. It considers three views that must be analyzed in

 ISBN: 978-1-61208-438-1

order to develop SOA. The Business and Information vision are inspired from the works of [9] [11] [13] [14] [16] [17]. SOA^{+d} method contribution consists on the proposal of the third vision which is: Decision.

SOA^{+d} follows a downward approach to discover the services. SOA^{+d} articulate around four phases: Analysis, Identification and categorize service, Services modeling and Realization pahse.

A. Phase 1: The analysis

includes three steps: information system analysis, business analysis and decision analysis steps [4].

B. Phase 2: Identify and categorize services

Is based on the cartographies already worked out to identify the services (business, information and decision services) [18]. As we already underlined, we identify three service types: services which exist on the business level, information system and decision services.

C. Phase 3: Services Modeling

In this step the services must be modeled with formalism. We adopt a specification at the base of the SoaML language [19] which offers a high level of abstraction, then it is necessary to refine the services to make them specific to a given platform.

D. Phase 4: The realization

Proposes to develop the services and to deploy them to be called upon. The technical choice (data base management system, development environment, application server, processes business Management system, etc.) must be done in this phase.

IV. Recruitment and Selection

This research is based on the process of Recruitment and Selection of SAAD DAHLAB University, located in the north region of Algeria. The recuitment process covers:

- Requesting a person with certain skills and abilities to fill a vacancy.
- Advertising the vacancy internal and external.
- Scheduling psych technical test, interviews, medical exams, etc.
- Collecting results of tests and interviews.
- Updating the candidates list.

A. Phase 1: The analysis phase

The Recruitment process begins when a Personnel Requisition is made. If the job description does not exist, it is created by a Human Resources Analyst. If the person who made the request does not have the level of authority, the process continues to approve request task.

The Recruitment process includes two sub processes :

- **Job Vacancy Advertisement Sub process** : The Human Resources area must arrange and place the advertisements in an appropriate medium. The advertisements can be placed internal or externally; the proposed architecture gives the flexibility to choose between them.
- **The selection process** : evaluates possible candidates for a vacancy; the sub process includes test and interview scheduling, enter their results and select the person.

Figure 1 shows the Recruitment business process with the BPMN (Business Process Management Notation) language.

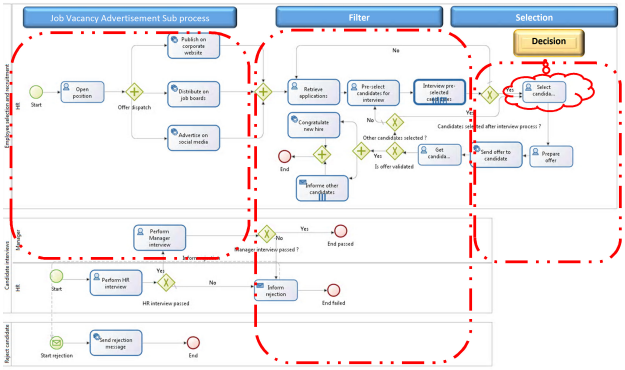


Figure 1. The Recruitment and Selection Process BPMN

B. Phase 2: Identify and categorize services

After the use of the analysis defined through the phase 1 to the Recruitment process, we found the services of Business, Information and Decision levels; this is shown in Figure 2.

The Business Process entity is Recruitment; it holds information about the personnel requisition such as Job Title, Number of vacancies needed, Area, and other information about the position. The entity is related to the Job Description, Advertisement and Candidates entities. The relationship between Recruitment and Candidates is from one of many; it is necessary to include several candidates in a Selection Process. The main attributes of the Job Description are: Title, Code, Responsibilities, Abilities, Experience and Job Description. The Advertisement entity includes Job Title, Location, Company Description, Contact Details, and Ideal Candidates. The Candidates entity includes Name, Last Name, CV file, Email.

Selecting a candidate is a complex problem involving qualitative and quantitative multi-criteria. The first step in any candidate rating procedure is to find the appropriate criteria to be used for assessing the candidate. To comply with the criteria for candidate selection and their importance, required data were collected.
In order to select the right candidate, the Analytic Hierarchy Process (AHP) [6] approach has been adopted. The AHP is a theory of measurement through pairwise comparisons and relies on the judgements of experts to derive priority scales. Figure 2 shows the services of decision level.

 ISBN: 978-1-61208-438-1

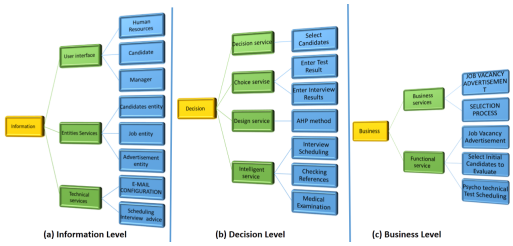


Figure 2. Decision level of Recruitment Process

C. Phase 3: Services Modeling

The mapping from BPMN diagram to SoaML model requires first and foremost a correspondence between the elements of BPMN and SoaML elements. For this, we use the mapping defined in the research work already done [7] [17] [20] and [21].

After performing the transformation rules we obtain the services modeling of Recruitment Process illustrate in Figure 3 and Figure 4.

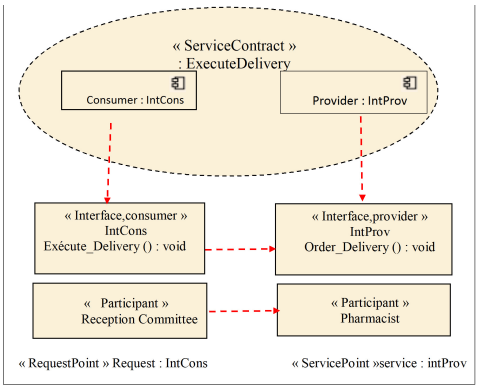


Figure 3. SoaML contratdiagram

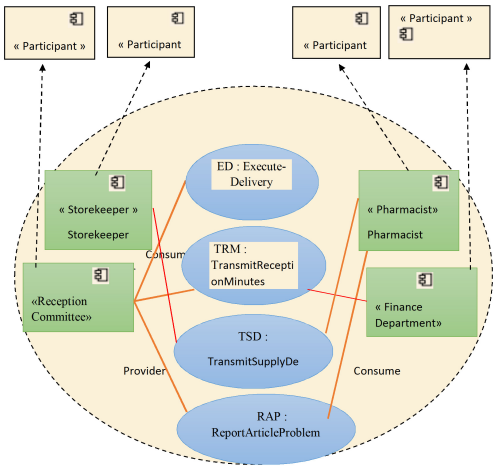


Figure 4. SoaML architecture diagram

D. Phase 4: The realization

Figure 5 summarizes the Services Integration in Recruitment Process.

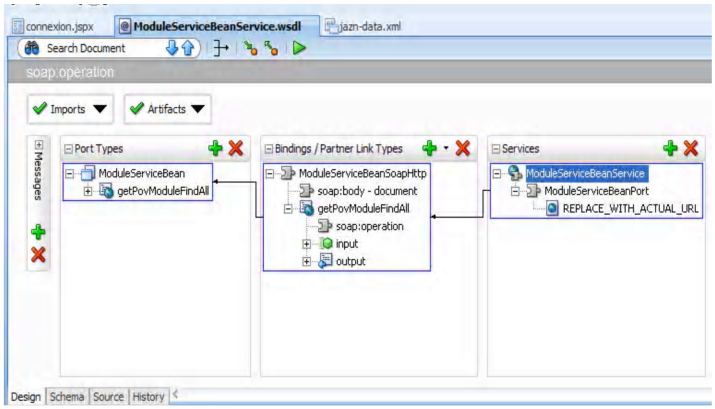


Figure 5. Services Integration

As shows Figure 6, during the Selection Process a Candidate must attend several interviews with different people. For each interview it is necessary to include the results.

UNIVERSITE SAAD DAHLAB-BLIDA
Faculté des Sciences
Laboratoire LRDSI

Liste finale de candidats pour le poste de directeur			
Nom	**Serge**	**Jean**	**Marc**
Âge	40 ans	45 ans	55 ans
Expérience pertinente	15 années	20 années	30 années
Niveau d'éducation	Baccalauréat	Maîtrise	Baccalauréat
Appui de la direction	Très bon	Bon	Bon

Matrice des critères de décisions en % relatifs					
	Âge	**Expérience**	**Éducation**	**Appui**	**Moyenne**
Âge	0.133	0.128	0.182	0.130	0.143
Expérience	0.533	0.513	0.455	0.522	0.506
Éducation	0.067	0.103	0.091	0.087	0.087
Appui	0.267	0.256	0.273	0.261	0.264

Figure 6. Preselection

Afterwards, the person who made the requisition must select the final candidate to fill the vacancy. As shows Figure 7, if the selected candidate accepts a salary offer the sub process ends.

UNIVERSITE SAAD DAHLAB-BLIDA
Faculté des Sciences
Laboratoire LRDSI

	Âge	Expérience	Éducation	Appui
Serge	0.539	0.164	0.2	0.6
Jean	0.297	0.297	0.6	0.2
Marc	0.164	0.539	0.2	0.2

*

Moyenne	
Âge	0.143
Expérience	0.506
Éducation	0.087
Appui	0.264

=

Candidats	
Serge	0.336
Jean	0.298
Marc	0.366

Matrice de solutions				
	Âge	**Expérience**	**Éducation**	**Appui**
Serge	0.539	0.164	0.2	0.6
Jean	0.297	0.297	0.6	0.2
Marc	0.164	0.539	0.2	0.2

Figure 7. Select the final candidate

 ISBN: 978-1-61208-438-1

V. Conclusion and Future Work

The increasingly globalized world necessitates to choose the best employee for the company, otherwise, both sides have to overcome several losses. This project was prepared with the aim to shed light on Blida University about effective recruitment process.

In this paper, we developed a system that works automatically to find the most suitable candidate for vacancy according to AHP model and using the new standard DMN. The various contributions carried out in our work are summarized as follow:

- **Introduction of the decision aspect into the SOA Services Oriented Architecture** The new approach, that we developed, extended the principles of the SOA on the totality of the company system. It brought new concepts and it restructures the company architecture in a manner that it is nimbler and able to take part in decisions from a request.
- **Proposed solution based on standards languages and notation**: in this paper we have presented a detailed approach using existing modeling languages such as UML, BPMN and SOAML. The proposed approach helps in specification, design and realization of a new type of service to depict the decision components in SOA.
- **A new approach of SOA uses DMN**: SOA architecture, that is developed is the first architecture of SOA that considers the modeling of the decision making aspect by the use of DMN; it extends the principles of the SOA , on the totality of the entrprise system. The decision introduced in SOA^{+d} is defined by the use of DMN. It is worth to note here that the suggested and modelled decision in SOA^{+d} contains more information than SOA approaches proposed in literature. As it was mentioned previously, the decision model that is obtained from the DMN notation is complete and shows how to automate the decision.

In our future work, we envisage the tool developement for proposed architecture to obtain a framework. Moreover, we are working on the implementation of the mapping rules defined within the framework, and more specifically those that allow us to obtain (as automatically as possible) services details from real Computational Independent models. Moreover, since our approach follows an MDE approach for the Service Oriented Development of SOA, we are currently working on the code generation from the models for different Web Services platforms. In the light of all these, we shall be able to complete the integration process between high level SOA and the decision implementation.

References

[1] N. Djenni.Rezoug, F. Nader, and F. Boumahdi, "A new approach to supporting runtime decision making in mobile olap," International Journal of Information and Communication Technology, 2015 In press.

[2] J. Taylor, A. Fish, J. Vanthienen, and P. Vincent, "Emerging standards in decision modeling," BPM and Workflow Handbook series, 2013.

[3] M. Thorpe, J. Holm, G. van den Boer et al., Discovering the Decisions within Your Business Processes using IBM Blueworks Live. IBM Redbooks, 2014.

[4] F. Boumahdi, R. Chalal, A. Guendouz, and K. Gasmia, "Soa+d: a new way to design the decision in soabased on the new standard decision model and notation (dmn)," Service Oriented Computing and Applications, 2014, pp. 1–19. [Online]. Available: http://dx.doi.org/10.1007/s11761-014-0162-x

[5] O. DMN, "Decision modeling notation," OMG, http://www.omg.org/spec/DMN/1.0/Beta1/PDF, Tech. Rep., 2014.

[6] T. L. Saaty, "Decision making with the analytic hierarchy process," International journal of services sciences, vol. 1, no. 1, 2008, pp. 83–98.

[7] J. Amsden, "Modeling with soaml, the service-oriented architecture modeling language." IBM, Tech. Rep., 2010.

[8] R. Börner and Goeken, "Identification of business services," in 15th Americas Conference on Information Systems (AMCIS), 2010.

[9] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "Soma: A method for developing service-oriented solutions," IBM systems Journal, vol. 47, no. 3, 2008, pp. 377–396.

[10] A. T. Rahmani, V. Rafe, S. Sedighian, and A. Abbaspour, "An mda-based modeling and design of service oriented architecture," in Computational Science–ICCS 2006. Springer, 2006, pp. 578–585.

[11] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[12] B. Berkem, "From the business motivation model (bmm) to service oriented architecture (soa)." Journal of Object Technology, vol. 7, no. 8, 2008, pp. 57–70.

[13] M. P. Papazoglou and W.-J. Van Den Heuvel, "Service-oriented design and development methodology," International Journal of Web Engineering and Technology, vol. 2, no. 4, 2006, pp. 412–442.

[14] S. Chaari, F. Biennier, J. Favrel, and C. Benamar, "Towards a service-oriented enterprise based on business components identification," in Enterprise Interoperability II. Springer, 2007, pp. 495–506.

[15] K. Mittal, "Build your soa, part 3: The service-oriented unified process," IBM developer Works, www.ibm. com/developerworks/library/ws-soa-method1.html, Tech. Rep., 2006.

[16] V. De Castro, E. Marcos, and J. M. Vara, "Applying cim-to-pim model transformations for the service-oriented development of information systems," Information and Software Technology, vol. 53, no. 1, 2011, pp. 87–105.

[17] C. Casanave, "Enterprise service oriented architecture using the omg soaml standard, a model driven solutions," ModelDriven.org, Tech. Rep., 2012.

[18] F. Boumahdi and R. Chalal, "Extending the service oriented architecture to include a decisional components," in Intelligent Decision Technology Support in Practice, ser. Smart Innovation, Systems and Technologies, J. W. Tweedale, R. Neves-Silva, L. C. Jain, G. Phillips-Wren, J. Watada, and R. J. Howlett, Eds. Springer International Publishing, 2016, vol. 42, pp. 185–199. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-21209-8_11

[19] S. OMG, "Service oriented architecture modeling version 1.0.1." OMG, http://http://www.omg.org/spec/SoaML/1.0.1/PDF, Tech. Rep., 2012.

[20] B. Elvesæter, A.-J. Berre, and A. Sadovykh, "Specifying services using the service oriented architecture modeling language (soaml)-a baseline for specification of cloud-based services." in CLOSER, 2011, pp. 276–285.

[21] B. Elvesaeter, D. Panfilenko, S. Jacobi, and C. Hahn, "Aligning business and it models in service-oriented architectures using bpmn and soaml," in Proceedings of the First International Workshop on Model-Driven Interoperability. ACM, 2010, pp. 61–68.

[22] B. OMG, "Business process modeling notation (bpmn)," OMG, http://www.omg.org/spec/BPMN/2.0/PDF, Tech. Rep., 2011.

[23] F. Boumahdi and R. Chalal, "Soada: A new architecture to enrich soa with a decisional aspect," International Journal of Systems and Service-Oriented Engineering (IJSSOE), vol. 4, no. 2, 2014, pp. 13–27.

 ISBN: 978-1-61208-438-1